Le traitement des commandes dans une partition conversationnelle Fortran sur IRIS 50

Jean-Claude Chupin

N° d'ordre:

THESE

présentée à

L'UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE

pour obtenir

LE GRADE DE DOCTEUR-INGENIEUR

par

Jean-Claude CHUPIN

Ingénieur I.P.G.

LE TRAITEMENT DES COMMANDES DANS UNE PARTITION CONVERSATIONNELLE FORTRAN SUR IRIS 50

Thèse soutenue le 10 Juillet 1971 devant la Commission d'Examen:

Monsieur	N. GASTINEL	Président
Messieurs	C. BOLLIET	Examinateurs
	R. BOUCHE	
	M. GRIFFITHS	

UNIVERSITE SCIENTIFIQUE
ET MEDICALE DE GRENOBLE

LISTE DES PROFESSEURS

Président : Monsieur Michel SOUTIF
Vice-Président : Monsieur Gabriel CAU

PROFESSEURS TITULAIRES

MM.	ANGLES D'AURIAC Paul	Mécanique des fluides
	ARNAUD Georges	Clinique des maladies infectieuses
	ARNAUD Paul	Chimie
	AYANT Yves	Physique approfondie
	BARBIER Jean-Claude	Physique expérimentale
	BARBIER Reynold	Géologie appliquée
	BARJON Robert	Physique nucléaire
	BARNOUD Fernand	Biosynthèse de la cellulose
	BARRIE Joseph	Clinique chirurgicale
	BENOIT Jean	Radioélectricité
	BESSON Jean	Electrochimie
	BEZES Henri	Chirurgie générale
	BLAMBERT Maurice	Mathématiques pures
	BOLLIET Louis	Informatique (IUT B)
	BONNET Georges	Electrotechnique
	BONNET Jean-Louis	Clinique ophtalmologique
	BONNET-EYMARD Joseph	Pathologie médicale
	BONNIER Etienne	Electrochimie Electrométallurgie
	BOUCHERLE André	Chimie et Toxicologie
	BOUCHEZ Robert	Physique nucléaire
	BRAVARD Yves	Géographie
	BRISSONNEAU Pierre	Physique du Solide
	BUYLE-BODIN Maurice	Electronique
	CABANAC Jean	Pathologie chirurgicale
	CABANEL Guy	Clinique rhumatologique et hydrologique
	CALAS François	Anatomie
	CARRAZ Gilbert	Biologie animale et pharmacodynamie
	CAU Gabriel	Médecine légale et Toxicologie
	CAUQUIS Georges	Chimie organique
	CHABAUTY Claude	Mathématiques pures
	CHATEAU Robert	Thérapeutique
	CHENE Marcel	Chimie papetière
	COEUR André	Pharmacie chimique
	CONTAMIN Robert	Clinique gynécologique
	COUDERC Pierre	Anatomie Pathologique
	CRAYA Antoine	Mécanique
Mme	DEBELMAS Anne-Marie	Matière médicale
MM.	DEBELMAS Jacques	Géologie générale
	DEGRANGE Charles	Zoologie
	DESSAUX Georges	Physiologie animale
	DODU Jacques	Mécanique appliquée
	DREYFUS Bernard	Thermodynamique
	DUCROS Pierre	Cristallographie
	DUGOIS Pierre	Clinique de Dermatologie et Syphiligraphie

	FAU René	Clinique neuro-psychiatrique
	FELICI Noël	Electrostatique
	GAGNAIRE Didier	Chimie physique
	GALLISSOT François	Mathématiques pures
	GALVANI Octave	Mathématiques pures
	GASTINEL Noël	Analyse numérique
	GERBER Robert	Mathématiques pures
	GIRAUD Pierre	Géologie
	KLEIN Joseph	Mathématiques pures
Mme	KOFLER Lucie	Botanique et physiologie végétale
MM.	KOSZUL Jean-Louis	Mathématiques pures
	KRAVTCHENKO Julien	Mécanique
	KUNTZMANN Jean	Mathématiques appliquées
	LACAZE Albert	Thermodynamique
	LACHARME Jean	Biologie végétale
	LATURAZE Jean	Biochimie pharmaceutique
	LEDRU Jean	Clinique médicale B
	LLIBOUTRY Louis	Géophysique
	LOUP Jean	Géographie
Mle	LUTZ Elisabeth	Mathématiques pures
MM.	MALGRANGE Bernard	Mathématiques pures
	MALINAS Yves	Clinique obstétricale
	MARTIN-NOEL Pierre	Séméiologie médicale
	MASSEPORT Jean	Géographie
	MAZARE Yves	Clinique médicale A
	MICHEL Robert	Minéralogie et Pétrographie
	MOURIQUAND Claude	Histologie
	MOUSSA André	Chimie nucléaire
	NEEL Louis	Physique du Solide
	OZENDA Paul	Botanique
	PAUTHENET René	Electrotechnique
	PAYAN Jean-Jacques	Mathématiques pures
	PEBAY-PEYROULA Jean-Claude	Physique
	PERRET René	Servomécanismes
	PILLET Emile	Physique industrielle
	RASSAT André	Chimie systématique
	RENARD Michel	Thermodynamique
	REULOS René	Physique industrielle
	RINALDI Renaud	Physique
	ROGET Jean	Clinique de pédiatrie et de puériculture
	SANTON Lucien	Mécanique
	SEIGNEURIN Raymond	Microbiologie et Hygiène
	SENGEL Philippe	Zoologie
	SILBERT Robert	Mécanique des fluides
	SOUTIF Michel	Physique générale
	TANCHE Maurice	Physiologie
	TRAYNARD Philippe	Chimie générale
	VAILLAND François	Zoologie
	VAUQUOIS Bernard	Calcul électronique
Mme	VERAIN Alice	Pharmacie galénique
	VERAIN André	Physique
Mme	VEYRET Germaine	Géographie
MM.	VEYRET Paul	Géographie
	VIGNAIS Pierre	Biochimie médicale
	YOCCOZ Jean	Physique nucléaire théorique

PROFESSEURS ASSOCIES

MM.	BULLEMER Bernhard	Physique
	RADHAKRISHNA Pidatala	Thermodynamique

PROFESSEURS SANS CHAIRE

MM.	AUBERT Guy	Physique
Mme	BARBIER Marie-Jeanne	Electrochimie
MM.	BARRA Jean	Mathématiques appliquées
	BEAUDOING André	Pédiatrie
	BERTRANDIAS Jean-Paul	Mathématiques appliquées
	BIAREZ Jean-Pierre	Mécanique
	BONNETAIN Lucien	Chimie minérale
Mme	BONNIER Jane	Chimie générale
MM.	CARLIER Georges	Biologie végétale
	COHEN Joseph	Electrotechnique
	COUMES André	Radioélectricité
	DEPASSEL Roger	Mécanique des Fluides
	DEPORTES Charles	Chimie minérale
	DESRE Pierre	Métallurgie
	DOLIQUE Jean-Michel	Physique des plasmas
	GAUTHIER Yves	Sciences biologiques
	GEINDRE Michel	Electroradiologie
	GIDON Paul	Géologie et Minéralogie
	GLENAT René	Chimie organique
	HACQUES Gérard	Calcul numérique
	JANIN Bernard	Géographie
Mme	KAHANE Josette	Physique
MM.	LATREILLE René	Chirurgie générale
	LAURENT Pierre	Mathématiques appliquées
	MULLER Jean-Michel	Thérapeutique
	PERRIAUX Jean-Jacques	Géologie et minéralogie
	POULOUJADOFF Michel	Electrotechnique
	REBECQ Jacques	Biologie (CUS)
	REVOL Michel	Urologie
	REYMOND Jean-Charles	Chirurgie générale
	ROBERT André	Chimie papetière
	SARRAZIN Roger	Anatomie et chirurgie
	SARROT-REYNAULD Jean	Géologie
	SIBILLE Robert	Construction Mécanique
	SIROT Louis	Chirurgie générale
Mme	SOUTIF Jeanne	Physique générale
M.	VALENTIN Jacques	Physique nucléaire

MAITRES DE CONFERENCES ET MAITRES DE CONFERENCES AGREGES

Mle	AGNIUS-DELORD Claudine	Physique pharmaceutique
	ALARY Josette	Chimie analytique
MM.	AMBLARD Pierre	Dermatologie
	AMBROISE-THOMAS Pierre	Parasitologie
	ARMAND Yves	Chimie

	BEGUIN Claude	Chimie organique
	BELORIZKY Elie	Physique
	BENZAKEN Claude	Mathématiques appliquées
Mme	BERTRANDIAS Françoise	Mathématiques pures
MM.	BLIMAN Samuel	Electronique (EIE)
	BLOCH Daniel	Electrotechnique
Mme	BOUCHE Liane	Mathématiques (CUS)
MM.	BOUCHET Yves	Anatomie
	BOUSSARD Jean-Claude	Mathématiques appliquées
	BOUVARD Maurice	Mécanique des Fluides
	BRIERE Georges	Physique expérimentale
	BRODEAU François	Mathématiques (IUT B)
	BRUGEL Lucien	Energétique
	BUISSON Roger	Physique
	BUTEL Jean	Orthopédie
	CHAMBAZ Edmond	Biochimie médicale
	CHAMPETIER Jean	Anatomie et organogénèse
	CHARACHON Robert	Oto-Rhino-Laryngologie
	CHIAVERINA Jean	Biologie appliquée (EFP)
	CHIBON Pierre	Biologie animale
	COHEN-ADDAD Jean-Pierre	Spectrométrie physique
	COLOMB Maurice	Biochimie médicale
	CONTE René	Physique
	CROUZET Guy	Radiologie
	DURAND Francis	Métallurgie
	DUSSAUD René	Mathématiques (CUS)
Mme	ETERRADOSSI Jacqueline	Physiologie
MM.	FAURE Jacques	Médecine légale
	GAVEND Michel	Pharmacologie
	GENSAC Pierre	Botanique
	GERMAIN Jean-Pierre	Mécanique
	GIDON Maurice	Géologie
	GRIFFITHS Michael	Mathématiques appliquées
	GROULADE Joseph	Biochimie médicale
	HOLLARD Daniel	Hématologie
	HUGONOT Robert	Hygiène et médecine préventive
	IDELMAN Simon	Physiologie animale
	IVANES Marcel	Electricité
	JALBERT Pierre	Histologie
	JOLY Jean-René	Mathématiques pures
	JOUBERT Jean-Claude	Physique du Solide
	JULLIEN Pierre	Mathématiques pures
	KAHANE André	Physique générale
	KUHN Gérard	Physique
Mme	LAJZEROWICZ Jeannine	Physique
MM.	LAJZEROWICZ Joseph	Physique
	LANCIA Roland	Physique atomique
	LE JUNTER Noël	Electronique
	LEROY Philippe	Mathématiques
	LOISEAUX Jean-Marie	Physique nucléaire
	LONGEQUEUE Jean-Pierre	Physique nucléaire
	LUU DUC Cuong	Chimie organique
	MACHE Régis	Physiologie végétale
	MAGNIN Robert	Hygiène et Médecine préventive
	MARECHAL Jean	Mécanique
	MARTIN-BOUYER Michel	Chimie (CUS)
	MAYNARD Roger	Physique du Solide
	MICOUD Max	Maladies infectieuses
	MOREAU René	Hydraulique (INP)

	NEGRE Robert	Mécanique
	PARAMELLE Bernard	Pneumologie
	PECCOUD François	Analyse (IUT B)
	PEFFEN René	Métallurgie
	PELMONT Jean	Physiologie animale
	PERRET Jean	Neurologie
	PERRIN Louis	Pathologie expérimentale
	PFISTER Jean-Claude	Physique du Solide
	PHELIP Xavier	Rhumatologie
Mle	PIERY Yvette	Biologie animale
	RACHAIL Michel	Médecine interne
	RACINET Claude	Gynécologie et obstétrique
	RICHARD Lucien	Botanique
Mme	RINAUDO Marguerite	Chimie macromoléculaire
MM.	ROMIER Guy	Mathématiques (IUT B)
	ROUGEMONT (DE) Jacques	Neuro-chirurgie
	STIEGLITZ Paul	Anesthésiologie
	STOEBNER Pierre	Anatomie pathologique
	VAN CUTSEM Bernard	Mathématiques appliquées
	VEILLON Gérard	Mathématiques appliquées (INP)
	VIALON Pierre	Géologie
	VOOG Robert	Médecine interne
	VROUSSOS Constantin	Radiologie
	ZADWORNY François	Electronique

MAITRES DE CONFERENCES ASSOCIES

MM.	BOUDOURIS Georges	Radioélectricité
	CHEEKE John	Thermodynamique
	GOLDSCHMIDT Hubert	Mathématiques
	YACOUD Mahmoud	Médecine légale

CHARGES DE FONCTIONS DE MAITRES DE CONFERENCES

Mme	BERIEL Hélène	Physiologie
Mme	RENAUDET Jacqueline	Microbiologie

A ma femme et à Stéphane notre fille ...

Je tiens à remercier particulièrement :

Monsieur le Professeur N. GASTINEL, *qui m'a fait l'honneur de présider le jury,*

Monsieur le Professeur L. BOLLIET, *responsable du contrat, pour ses précieux conseils dans de multiples domaines et pour la direction de cette thèse,*

Monsieur R. BOUCHE, *qui a bien voulu s'intéresser à mon travail et a accepté de le juger.*

J'adresse également mes remerciements à Messieurs DELPUECH, AVON *et* GIRES *avec lesquels nous avons formé pendant quinze mois une équipe de travail très agréable.*

Je remercie enfin Madame J. CARRY *pour sa diligence et son excellente humeur, ainsi que tout le personnel de l'I.M.A.G. qui a permis la production de ce document.*

PREFACE

A l'origine de ce travail, se place un contrat passé en 1969 entre l'I.M.A.G. et la C.I.I. pour la réalisation d'une partition conversationnelle FORTRAN sur IRIS 50. Le sujet de cette thèse est le traitement des commandes dans le cadre de ce système. (1).

Je désire indiquer l'esprit dans lequel ces pages ont été écrites. Lorsque la réalisation d'une partition conversationnelle FORTRAN sur IRIS 50 a été demandée, les exigences et les contraintes étaient extrêmement faibles. Si cela donnait toute liberté et initiative, cela obligeait également à motiver toutes les décisions qui allaient être prises quant au contenu de cette partition. Il allait donc y avoir une longue phase de conception et de définition au cours de laquelle les buts et les objectifs allaient s'élaborer et prendre leur forme définitive. Cette période très enrichissante qui se situe avant toute programmation, correspond à deux activités distinctes :

- la traduction sous forme d'objectifs d'une certaine idée que le concepteur se fait du produit dont il a la responsabilité. Ceci aboutit à des spécifications de définition dont la version définitive est le fruit des discussions avec le destinataire du produit.

(1) L'ordinateur IRIS 50 muni de son système SIRIS II peut comporter trois partitions : une partition série, une partition temps réel, une partition parallèle; dans l'état actuel la partition Fortran est la partition parallèle. Pour avoir une vision complète de la partition, les pages qui suivent sont à associer avec les travaux de Monsieur DELPUECH pour l'organisation générale du système et la partie moniteur local. La partie analyse génération FORTRAN a été confiée à Messieurs AVON et GIRES. Elle est contenue dans un rapport de projet de fin d'étude de l'I.P.G. soutenu en 1970 et guidé par Monsieur DELPUECH et moi-même.
Parallèlement à cette réalisation une simulation du système a été développée; elle fait l'objet de la thèse de 3° Cycle soutenue par Mademoiselle Joëlle COUTAZ à l'I.M.A.G. en 1970.

- l'étude des méthodes qui seront employées pour implémenter ces objectifs. La description de ces méthodes constitue les spécifications de réalisation. Elles correspondent à la résolution des problèmes de réalisation à tous les niveaux.

Une période très importante est celle de la transition entre ces deux activités car c'est elle qui assure l'homogénéité de l'ensemble : elle consiste en l'adoption d'une optique de réalisation qui doit refléter les préoccupations antérieures tout en tenant compte des contraintes de programmation.

Or, un des soucis dominants était de privilégier au maximum les utilisateurs et de créer entre eux et la partition la plus grande 'intimité' possible. Cette préoccupation va marquer le travail bien au-delà de la définition du jeu de commandes et dans les plus bas niveaux de la réalisation. En effet, il est évident que plus on permet de facilités, plus la réalisation va croissante en complexité : le matériel de réalisation (langage de programmation, software, hardware) existant dont nous allons être tributaires risque d'obéir à des préoccupations contraires aux nôtres, créant des difficultés et des contraintes importantes.

Conscient des difficultés de réalisation qui allaient se présenter, j'ai essayé de mettre sur pied une philosophie et une stratégie de réalisation qui me permettent de satisfaire des objectifs qui me tenaient spécialement à coeur :

- clarté dans les spécifications
- modularité
- simplicité
- possibilité de se rendre compte le plus tôt possible au cours du travail des impossibilités obligeant à remettre en question des objectifs antérieurs. Ce dernier point m'apparaît fondamental.
- généralité enfin.

Ceci étant précisé, j'ai fait le choix suivant : plutôt que de présenter simplement la réalisation effective et de laisser au lecteur le soin de découvrir le fil conducteur, je présenterai d'abord la philosophie et la stratégie de réalisation adoptées avec leurs motivations, en évitant de faire figurer les détails de la programmation chaque fois qu'ils n'apporteront pas d'éclaircissements.

Toutefois afin de fixer complètement les idées et de permettre de juger de l'efficacité des principes adoptés, un chapitre sera consacré à la description d'un cas de réalisation complète.

ABREVIATIONS

PC : parition conversationnelle

FSC : fichier source-courant

FDC : fichier donnée courant

TI : Téléimprimeur

IL : Identificateur de ligne

ZCC: zone de contrôle commune

ZCL : zone de contrôle de ligne

MAE : machine à écrire

LPR : lecteur perforateur de ruban

BDC : bloc de décodage de la commande

BPC : bibliothèque privée conversationnelle

BCC : bibliothèque commune conversationnelle

BPB : bibliothèque privée batch

TDF : table de description de fichier

ZT : zone de travail

ZDP : zone des pages

ZNP : zone non partagée

RU : routine utilitaire

SPTC : sous-programme de traitement de commande

MTC : module de traitement de commande

TABLE DES MATIERES

SECONDE PARTIE : LES FICHIERS

INTRODUCTION

0 - INTRODUCTION GENERALE

0.1 Une présentation de la partition a priori

Il s'agit de réaliser un système conversationnel. Ceci implique une notion de partage entre un certain nombre d'utilisateurs. Le problème initial est donc celui de la synchronisation. Il est résolu par un élément du software de base de l'IRIS 50 : le SGT (système de gestion des transmissions). Le SGT permet la synchronisation de processus, notion dynamique définie à un instant donné par un descripteur de contrôle et le programme actif. (Il peut y avoir plusieurs descripteurs pour un même programme).

Dans ces conditions, la partition apparaît comme un ensemble de processus se synchronisant par un moniteur local (1) qui utilise le SGT. Les programmes constituant l'aspect spécifique du système seront des modules partageables réentrants constituant la partie dynamique des processus. Les processus ne possèdent pas la responsabilité de leurs synchronisations qui ne pourront avoir lieu que sur des requêtes implicites ou explicites au moniteur local.

Si on suppose que le moniteur local existe ainsi que les procédés de sa mise en oeuvre, la réalisation du système proposé apparaît alors de la façon suivante :

Réaliser de la façon la mieux adaptée possible un ensemble de programmes partageables réentrants que l'on considère comme un système a implanter sur une machine non 'nue'. Ces programmes seront activés par des commandes dont ils ignoreront l'auteur et permettront de faire fonctionner les différents éléments de la machine dans le but de réaliser certaines opérations.

La machine de base est un IRIS 50 muni de son software, c'est-à-dire du système d'exploitation SIRIS II complété par un moniteur local (1) utilisant le SGT.

(1) cf. Travaux de Monsieur DELPUECH.

Les spécifications générales du produit définissent (commandes) un certain nombre de fonctions nécessaires tandis que les caractéristiques de la 'machine' constituent le domaine des moyens possibles pour les satisfaire.

Sous cet éclairage, le produit à réaliser se décompose en deux parties :

- la réalisation des éléments à ajouter à la machine initiale pour la rendre apte à gérer une partition conversationnelle; c'est-à-dire un moniteur local alimenté par le moniteur d'enchaînement de SIRIS II, et les moyens d'utilisation de ce moniteur local. (1).

- la réalisation d'un sous-système correspondant au traitement des commandes et donnant à la partition son aspect spécifique. Les pages qui suivent concernent cette seconde partie.

0.2 Une certaine vision d'un système conversationnel

Qui dit conversationnel dit interaction entre des programmes et des utilisateurs disposant de terminaux. C'est cette interaction qui va retenir toute notre attention.

On désigne souvent par système conversationnel un certain enchaînement de fonctions réalisées par des composants classiques. L'aspect conversationnel réside dans la possibilité de réaliser l'enchaînement non plus automatiquement mais à la demande et sous la direction des utilisateurs. Bien que très intéressant, ce procédé ne permet une interaction qu'à un niveau assez global, tenant à la nature classique des composants.

Par contre, on introduit un domaine de possibilités très vaste, lorsqu'on modifie en plus de l'enchaînement la nature des composants. C'est ce qui nous occupera ici avec la réalisation d'un système destiné à faire fonctionner un compilateur FORTRAN incrémentiel et conversationnel.

Le parti sera pris délibérément d'adopter un point de vue utilisateur et de favoriser ce dernier au maximum en établissant un véritable dialogue. Dans ces conditions, on peut regarder un système conversationnel comme constitué :

(1) cf. Monsieur DELPUECH.

- d'une partie fonctionnelle incluant tout l'aspect spécifique (vocation du système) et la logique interne. Cette partie doit dans la mesure du possible être ignorée de l'utilisateur.

- d'une partie interaction qui sera portée à la connaissance des utilisateurs et qui comprendra les commandes d'une part et les messages envoyés par le système d'autre part.

Les commandes constituent le moyen par lequel les utilisateurs peuvent faire connaître leurs désirs et imposer leur volonté au système. Elles permettent de solliciter la partie spécifique : ce sont des demandes de services. La richesse et le choix du jeu de commandes permettent de juger de la puissance du système et de son adaptation aux buts fixés.

Les messages incarnent les réflexes du système et son niveau de conversation. Ils sont caractérisés par leur rapidité quand il s'agit de réponses et dans tous les cas par leurs contenus.

L'interaction, le dialogue présente deux aspects :

- un aspect simplificateur qui correspond à la possibilité de fournir le moins possible d'arguments au moment du lancement d'une commande et à laisser à la partition le soin de demander les arguments qui lui sont nécessaires à l'instant où ils lui sont nécessaires. On y retrouve le souci d'alléger le travail à la charge de l'utilisateur.

- un aspect contrôle qui correspond à la possibilité d'intervenir sur le déroulement des opérations (arrêt, redémarrage, demande de l'état d'avancement de la fonction, etc ...). Cet aspect est lié comme nous allons le voir plus loin au degré de réentrance et de modularité des fonctions. On verra dans quelle mesure la logique interne est concernée.

Ces deux aspects bien que de niveaux différents sont naturellement complémentaires. Le premier guide l'utilisateur au fur et à mesure que la partition avance dans son travail et le second soumet ce travail à la volonté des utilisateurs.

Ces quelques considérations devraient permettre de comprendre plus aisément les choix qui ont été faits pour réaliser le traitement des commandes tant pour ce qui concerne l'aspect externe (syntaxe des commandes) que pour ce qui a trait au fonctionnement interne.

PREMIERE PARTIE

PRINCIPES ET METHODES DE REALISATION

1 - UNE PHILOSOPHIE GENERALE DE REALISATION

1.1 Considérations générales

Le système qui était commandé représentait un travail important aussi bien par la taille (1) que par la nouveauté (Incrémentiel) et la diversité des problèmes à étudier (pagination, debug Fortran, multitasking, etc ...). Il était impensable dans ces conditions de procéder au hasard sans une méthode qui définisse et guide ensuite les différentes étapes.

Cette méthode restait à définir et il fallait choisir les concepts de base qui la soutiendraient.

La nature du produit et du destinataire ont fait retenir les concepts de spécification de modularité et de généralité.

Le contenu des spécifications pour une application nouvelle doit permettre la compréhension non seulement du fonctionnement mais des motivations du concepteur. Elles jouent un rôle fondamental pour un produit susceptible de modifications, et devant faire l'objet d'une maintenance.

Les spécifications peuvent suivant leur nature permettre au destinataire d'acquérir une compétence dans le domaine du produit.

La modularité traduit une démarche analytique et un souci de rendre indépendants les différents modules en vue de la définition, de la programmation et des tests. Elle permet les modifications avec le minimum de rejaillissements et de discontinuité. La modularité naîtra d'une analyse correcte des fonctions et de leurs relations. (le terme de relation désigne aussi bien les interfaces hardware que software).

La généralité est le concept qui permet un changement de vocation et des extensions avec le minimum de modifications du produit initial.

(1) Le problème de la taille était accentué par le nombre de personnes dans notre équipe : deux ingénieurs et deux élèves de 3ème Année de l'I.P.G.

Les deux derniers concepts, généralité et modularité sont étroitement dépendants car la modularité est une des conditions impératives de la généralité. Cette remarque conduit d'ailleurs à étendre la notion de modularité qui ne s'applique la plupart du temps qu'à la programmation et à l'aspect fonctionnel. Il semble au contraire qu'il faille faire jouer la modularité à un niveau beaucoup plus général comme le montre la remarque suivante :

Si on regarde le traitement des commandes dans une optique simplement fonctionnelle, il apparaît comme un ensemble de fonctions principales correspondant à peu près aux commandes qui se décomposent en un certain nombre de sous-fonctions. Ceci est le résultat de l'habitude d'une programmation à un seul niveau qui masque les niveaux véritables auxquels se situent les fonctions. (Par exemple, le niveau du dialogue ne se détache pas clairement). Même si la modularité est réalisée elle n'assure pas les remises en cause minimum si une fonction contient des éléments de niveaux différents : par exemple, si on met le "test du Break" dans un sous-programme qui réalise une autre fonction logique (même si elle présente un lien avec le Break), une modification du système de break oblige à reprendre la fonction dans son ensemble bien que la modification ne présente pas un caractère logique. On a eu le tort de mélanger le niveau physique et le niveau logique.

Il semble plus satisfaisant de faire l'approche suivante : on essaie de déterminer le niveau auquel se situent les différentes fonctions que nous utiliserons et on procède ensuite à l'implémentation de ces niveaux.

L'effort se portera donc non plus sur la réalisation d'une fonction mais sur la définition et la réalisation d'une certain nombre de niveaux. Si cette étude est correctement menée on ne courera pas le risque d'essayer de réaliser une certaine fonction tant bien que mal avec des moyens mal adaptés. Si, en outre, on réalise une certaine modularité entre les niveaux, on tend vers la généralité.

Les considérations qui viennent d'être développées concernent la modularité minimum. Il existe d'autres impératifs qui comme nous le verrons plus loin conduisent à augmenter le degré de modularité (l'automatisme du fonctionnement des commandes et la nécessité de segmenter les programmes sont deux de ces impératifs).

Un des critères secondaires qui préside à la définition d'une méthode de réalisation est la prise de conscience de l'importance de la phase de programmation et de tests. On décide de prendre en compte dès le début la répartition des travaux à effectuer. La méthode doit permettre à partir des spécifications de réalisation qui la concrétisent de procéder à une répartition des tâches qui s'accorde avec les moyens dont on dispose, tant sur le plan humain que sur le plan du matériel.

Ces quelques principes abstraits étant définis, il faut les traduire au niveau informatique.

1.2 Les éléments de traduction de concepts

Nous avons parlé de niveaux. C'est une notion très importante que nous allons préciser indépendamment de toute préoccupation de programmation.

1.2.1 Définition d'un niveau

Un niveau est constitué de 'machines' et est défini quant à sa position par un certain critère. Ce critère peut être par exemple le degré de sophistication ou d'indépendance vis-à-vis d'un élément (le domaine de la machine physique par exemple) mais aussi bien tout autre critère permettant de hiérarchiser les machines. Le nombre des machines à l'intérieur d'un niveau peut être quelconque. Il est lié à la pluralité des objectifs des niveaux supérieurs.

Définition d'une machine

Une machine est un ensemble de fonctions qui sont ses composants. Les fonctions peuvent ou non être indépendantes et on peut entrer dans la machine par certaines de ces fonctions. Elles seront fabriquées à l'aide de matériaux dont la nature dépend en général du niveau de la machine. (le matériau peut être un ensemble d'instructions pour une machine logique ou des composants électroniques pour une machine physique).

Définition d'une fonction

C'est un ensemble d'opérations qui permettent de réaliser un objectif.

Relations entre les machines

On doit distinguer deux aspects des relations : le plan logique qui traduit la nécessité de la relation et, englobe le contenu de la transmission, le plan physique qui réalise la relation de façon 'mécanique' et constitue la solution au problème posé par l'établissement de la dite relation. Les relations entre les machines de même niveau ne seront pas étudiées ici.

La définition des différentes relations définit le réseau des machines. Il dépend de la nature du système et de la philosophie de réalisation. Il a pour limite les moyens de communication entre les fonctions qui dépendent eux-mêmes du matériel utilisé pour fabriquer les fonctions.

Les contraintes imposées à ces notions

Elles l'ont été en fonction du produit particulier pour lequel nous utilisons ces différents concepts.

a) Un niveau ne comprend qu'une seule machine.

b) Les relations entre les machines correspondent à un empilement gigogne. Une machine ne peut communiquer qu'avec celle de niveau immédiatement inférieur ou immédiatement supérieur et pour les opérations suivantes :

- mise en oeuvre de la machine inférieure de haut en bas
- indication du fonctionnement (bon ou mauvais) de bas en haut.

1.2.2 L'utilisation des niveaux

Comme indiqué plus haut, les différents principes adoptés doivent permettre de guider la réalisation et les décisions.

Le concept de niveau servira à classer les fonctions dont on aura besoin. Le découpage en niveaux disjoints et la nature hiérarchique de l'empilement des niveaux opèrent un tri sur l'ensemble des fonctions. Il s'agit même d'une véritable relation d'équivalence : deux fonctions seront dites équivalentes si elles sont de même niveau.

La hiérarchie permet de définir une relation d'ordre sur les niveaux qui est la relation d'inégalité classique. Cette dernière relation sera très intéressante pour le partage du travail et le déroulement des tests. Il apparaît donc que le choix et le contenu des niveaux marquera la réalisation jusque dans ses moindres détails. La construction des niveaux est donc extrêmement importante.

1.2.3 Le choix des niveaux

L'idée de différents niveaux et les services que l'on peut attendre de cette notion sont relativement intuitifs. Le critère de choix est plus difficile à établir. Il a été effectué en tenant compte de difficultés que l'on présentait importantes et que l'on pensait pouvoir ainsi résoudre. Il s'agit essentiellement de soucis de clarté, et de progression en pas à pas dans la réalisation.

La modularité est également prise en compte dès ce stade.

On choisit un critère de détachement croissant vis-à-vis de la machine initiale (il s'agit,rappelons le, de l'IRIS 50 muni de SIRIS II et d'un moniteur local). Ce détachement est très lié à l'application particulière qui doit permettre le fonctionnement d'un compilateur Fortran incrémentiel. En effet, plus on montera dans la hiérarchie, plus on désirera mettre en oeuvre des fonctions spécialisées pour aboutir à la limite supérieure à des fonctions globales correspondant exactement aux commandes.

Compte tenu des contraintes sur l'empilement des niveaux, chaque niveau utilisera pour des besoins propres le niveau immédiatement inférieur. Le problème de l'ordre de construction des niveaux sera discuté plus loin dans la partie Stratégie de réalisation.

Nous allons maintenant nous intéresser à la liste des différents niveaux et à la description de leurs contenus. Nous situerons chaque niveau dans la hiérarchie et indiquerons ce qu'il permet de faire.

Nous retiendrons essentiellement 5 niveaux :

- le niveau -1 qui est celui du hardware et des microprogrammes traitant des profils binaires.
- le niveau 0 qui est le niveau physique mais sous sa forme utilisateur.

Il contiendra les opérations permettant de générer des profils binaires.

- le niveau 1 qui est le niveau physique évolué, c'est-à-dire celui qui permet de faire en une seule fois plusieurs opérations physiques élémentaires. Ses éléments comprendront en particulier des fonctions permettant de mettre en oeuvre toutes les fonctions du système existant et du moniteur local dont on pourra avoir besoin. C'est à ce niveau que seront résolus les problèmes de multitasking et de réentrance.

- le niveau 2 est celui des opérations fonctionnelles logiques. Ses éléments sont des fonctions liées à l'application particulière (Fortran). Leurs algorithmes traduisent pour la plupart les spécifications Fortran.

- le niveau 3 qui est celui du contrôle et de l'enchaînement. Les éléments en sont des fonctions globales qui correspondent en gros aux différentes commandes mises à la disposition de l'utilisateur.

Nous venons de voir la nature des éléments constitutifs des classes. Il faut maintenant présenter les communications entre les différents niveaux.

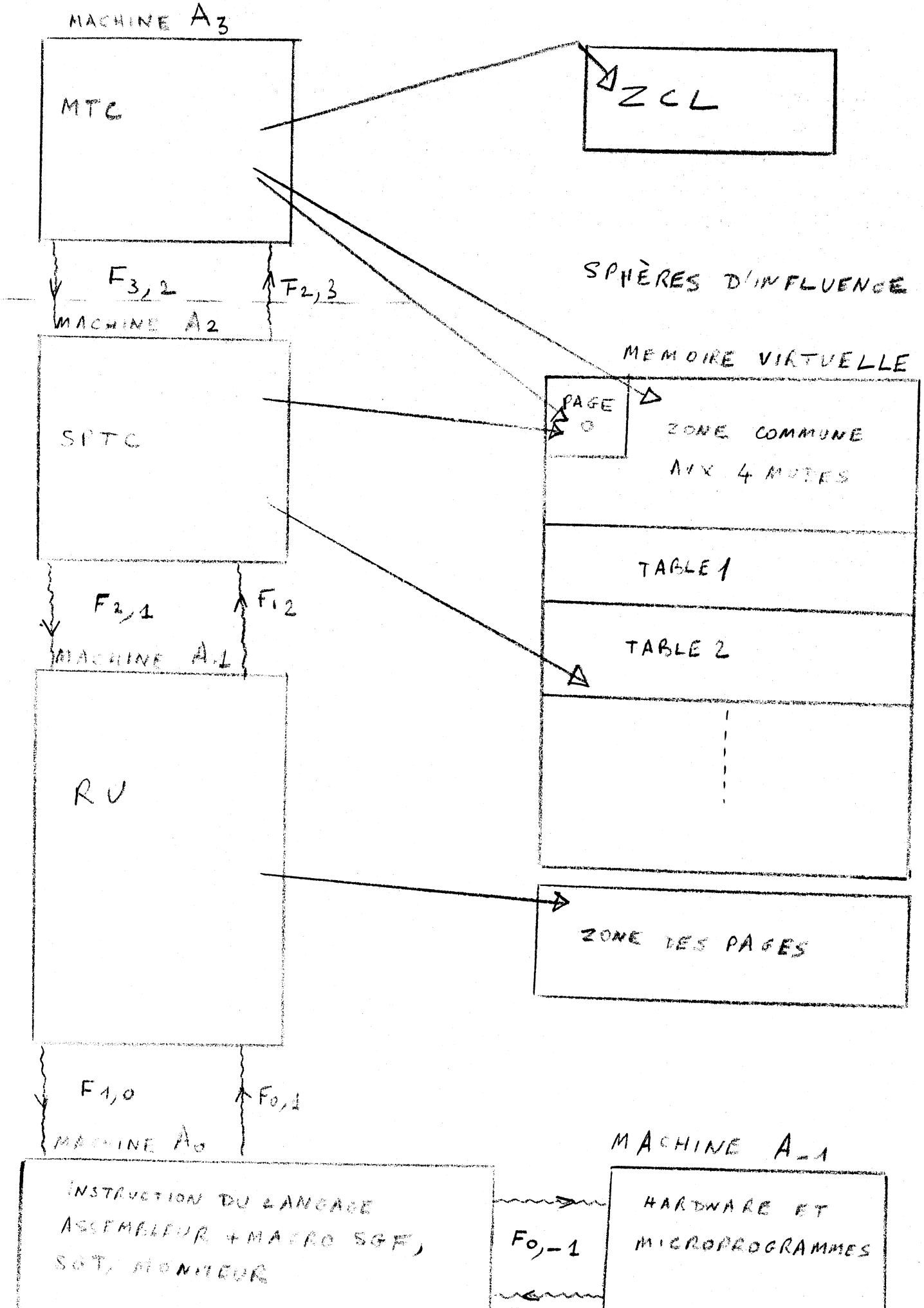
Si on isole une fonction, quelque soit son niveau, son déroulement consistera toujours à gérer des zones de travail. C'est-à-dire qu'après un certain nombre d'étapes, on procèdera à la mise en oeuvre d'une fonction de niveau -1. Il est donc nécessaire qu'une machine puisse utiliser les composants des machines de niveau inférieur et ce dans un ordre dont elle a l'entière responsabilité. On impose en plus que l'influence d'un niveau soit limitée au niveau immédiatement inférieur. On assiste à une demande de service de la part de la machine du plus haut niveau et il faut donc qu'une machine puisse communiquer des informations à celle du dessus et du dessous. Cette communication doit se faire par un canal propre aux deux machines concernées pour offrir le maximum de sécurité.

On a vu que les fonctions constituant une machine étaient indépendantes. Etant alors donnée une machine Mi on voit apparaître deux franges

de communication Fi, i-1 et Fi, i+1. Par ces franges qui sont des facultés extra fonctionnelles de la machine, une fonction de niveau i peut respectivement demander le service d'une fonction de niveau i-1 ou indiquer quelque chose à la fonction de la machine i+1 qui l'a appelée. On doit remarquer que les franges de communication doivent comprendre à la fois les zones de support de l'information transmise et le procédé de transmission.

Quant au processus de construction d'un niveau à partir des autres, il relève d'une stratégie de réalisation qui tient compte de la philosophie générale d'une part et des moyens de réalisation que l'on a adoptés (langage de programmation, effectifs, qualification, etc ...) d'autre part.

EMPILEMENT DES MACHINES ET SPHERES D'INFLUENCE



1.3 La programmation compte tenu de la philosophie générale

1.3.1 Le langage de programmation

Nous allons enfin nous intéresser aux problèmes de programmation. Commençons par le choix du langage :

- il doit refléter les niveaux
- il doit permettre de réaliser les objectifs fonctionnels que l'on s'est fixé
- il doit pouvoir rendre disponibles les fonctions du système existant
- il doit pouvoir fonctionner en multitasking (1).

Nous nous bornerons ici aux solutions qui nous étaient matériellement possibles. Ce qui aurait été le plus souhaitable sera discuté ultérieurement dans un paragraphe spécial traitant des éléments à offrir pour réaliser plus agréablement ou plus efficacement les objectifs qui étaient les nôtres. Il semble qu'un langage d'assemblage avec des facilités d'imbrication de macro soit assez bien adapté mais ceci nous était interdit pour plusieurs raisons :

- il n'existait pas de macro-assembleur
- il était demandé dans la mesure du possible de programmer en langage évolué (en l'occurence le Fortran).

Nous disposions donc du langage d'assemblage ASSIRIS et d'un Fortran sans extension que l'on puisse utiliser pour notre application particulière.

Nous allons donc passer en revue les trois niveaux en nommant les éléments et en indiquant quel langage a été choisi et pourquoi (2).

(1) cf. Monsieur DELPUECH pour le problème de l'utilisation du Fortran.

(2) On trouvera en Appendice n° 5 la liste des éléments constituant les niveaux 2 et 3.

1.3.2 Réalisation des niveaux

Niveau -1 : nous n'en sommes pas maîtres comme d'ailleurs chaque fois que l'on utilise une machine donnée.

Niveau 0 : c'est l'ensemble des instructions et macro-instructions d'ASSIRIS. On comprend également les macro-instructions SGF et SGT que nous sommes susceptibles d'utiliser.

Niveau 1 : c'est l'ensemble des Routines Utilitaires (RU). Les routines utilitaires sont des sous-programmes écrits en langage assembleur car on n'est pas encore à ce niveau détaché du niveau physique.

Niveau 2 : c'est l'ensemble de ce que nous appelons Sous-Programmes de Traitement de Commandes (SPTC). Nous utiliserons un langage plus évolué (Fortran) pour souligner l'aspect logique et pour indiquer l'indépendance prise vis-à-vis de la machine physique.

Niveau 3 : c'est l'ensemble des Modules de Traitement des Commandes. Faute de mieux, ils seront écrits en Fortran. Le changement de niveau, qui nous allons le voir existe bien, n'apparaît pas ici au niveau du langage de programmation , mais dans le type d'instruction Fortran utilisé.

1.3.3 Réalisation des communications internes

Par abus de langage nous parlerons de communications entre machines bien qu'il ait été plus correct de parler de communication entre deux fonctions de machines différentes. Avec les restrictions adoptées, nous utiliserons indifféremment le terme de machine ou de niveau. Entre les niveaux -1 et 0 il s'agit du processus de décodage et d'exécution d'une instruction machine par le hardware et les microprogrammes. Pour les autres niveaux, nous distinguerons encore l'aspect physique et l'aspect logique (contenu) de la transmission.

a) L'aspect physique :

C'est le mécanisme d'appel d'un sous-programme Fortran (CALL) de haut en bas et le mécanisme de retour (plus compliqué que le RETURN à cause du par-

tage entre plusieurs utilisateurs) de bas en haut. Le canal dont il a été question à propos des franges de communication est constitué par les paramètres formels.

b) L'aspect logique :

C'est le contenu de la transmission. Il est très important car il indique la quantité de responsabilité et la sphère d'influence mémoire que l'on accorde dynamiquement à la machine appelée.

Il y a en effet une grande différence entre la transmission d'une ligne source au compilateur et la transmission d'un pointeur sur la table des enchaînements qui conditionne toute une session (c'est le cas pour les SPTC de rangement de source depuis le tty). On peut dire que l'on n'accorde au compilateur qu'une confiance limitée.

Remarque :

Ceci est la présentation théorique de la frange de communication.

Un appel a deux significations :

- un changement de niveau
- une demande de service sur des informations que l'on transmet lors de l'appel.

La sécurité de la transmission pose des problèmes dans un contexte multi-utilisateur avec une programmation en Fortran (1).

Pour cette raison les informations ne seront passées par paramètres que dans des cas relativement restreints. Dans la majorité des cas, on passera les informations par la mémoire virtuelle qui assure la réentrance d'office en multi-utilisation. Ceci revient à définir un nouveau mécanisme de passage de paramètres.

(1) cf. Monsieur DELPUECH. Problèmes de réentrance du Fortran.

On ne doit cependant pas confondre frange de communication et zone de travail commune, au sens de la programmation. En effet, aussi bien dans le cas d'un appel classique avec paramètres que dans notre cas, le réalisateur n'est pas maître de la mise en commun des informations. Dans le cas classique, la responsabilité en est confiée au compilateur et dans notre cas, elle est décidée par le concepteur de la logique interne du système qui doit au moins dans ses décisions être distinct du programmeur. Dans ce dernier cas, la définition des zones d'influence des niveaux est fondamentale. Nous verrons plus loin comment elle sera utilisée systématiquement pour effectuer le partage du travail.

1.3.4 La réalisation du dialogue

Il s'agit d'indiquer comment on a implémenté le concept de dialogue dont il a été question dans l'introduction.

1.3.4.1 Le dialogue simplificateur

Il est réalisé par la possibilité offerte aux SPTC (Niveau 2) de demander des informations au téléimprimeur et d'y envoyer des messages. Ces opérations sont réalisées par deux routines utilitaires du moniteur local (1).

REC1(B, D, RES) Receive conversational

qui sort un identificateur de ligne et attend une réponse qu'il place dans le buffer en ZCL, en la faisant précéder de la longueur

SEND1

qui écrit au téléimprimeur les L caractères présents dans le buffer et précédés de leur longueur L.

(1) cf. Travaux de Monsieur DELPUECH.

1.3.4.2 Le dialogue de contrôle

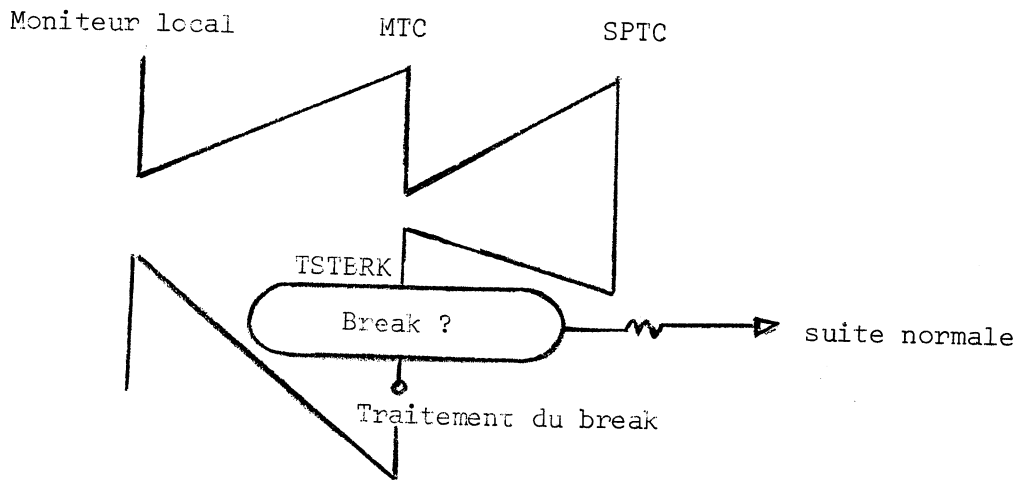
Il s'agit essentiellement de l'interruption d'un travail en cours et du redémarrage sous certaines conditions.

D'un point de vue logique, l'interruption ne peut se produire qu'en certains points dits "points de break". Il est donc inutile de s'apercevoir d'une interruption hors des points où il est possible de prendre des mesures pour la traiter. Le problème a été résolu par l'utilisation de la routine utilitaire du moniteur local TSTBRK(N). (1) qui permet de savoir si l'indicateur de Break a été positionné à un certain moment et non encore pris en compte.

Il est évident qu'une telle action relève du niveau 3 (MTC) et c'est par abus d'écriture que l'on rencontrera des appels de cette routine utilitaire dans le MTC.

Le traitement d'un break dépend du MTC qui l'effectue. Il donne lieu à des opérations spécifiques et se termine dans tous les cas par le passage en attente commande provoqué par un retour au "moniteur local".

Le schéma est le suivant :



Le redémarrage intervient en mode exécution et est mis en oeuvre par la commande REPARTIR (1).

(1) cf. Présentation des commandes.

Remarque :

Le niveau du dialogue rejaillit sur celui des SPTC. En effet, puisqu'on définit un certain nombre de points interruptibles, la taille et la fonction réalisées par un SPTC peuvent être modifiées. En particulier on peut être amené à éclater un SPTC en plusieurs pour créer de nouveaux points interruptibles.

1.3.5 Le traitement des erreurs

Il est dominé par les options suivantes :

- Une erreur doit pouvoir être détectée par une fonction de n'importe quel niveau.

- L'erreur ne doit donner lieu à une intervention de l'utilisateur que dans le cas où le travail ne peut pas continuer convenablement.

- C'est le niveau des MTC qui doit le signaler aux utilisateurs et demander une action.

Pour réaliser cela on crée le matériel suivant :

- Des répertoires de messages, l'un résident , l'autre sur disque (chaque message est repéré par son numéro).

- Une pile permettant à la fonction qui détecte d'empiler le n° du message.

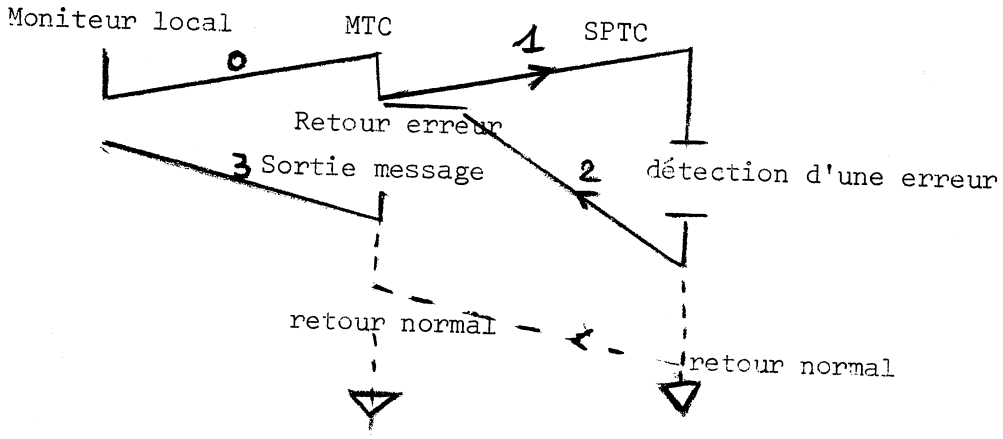
- Une routine utilitaire d'empilement du numéro.

- Une routine de sortie au TTI des messages dont les numéros sont dans la pile.

Cet ensemble est associé à un mécanisme d'appel d'un sous-programme de niveau inférieur comprenant deux adresses de retour (1) :

- l'une pour un traitement sans erreur
- l'autre pour le retour erreur.

Le mécanisme est le suivant :



1.3.6 Quelques problèmes de réalisation

Il s'agit de problèmes qui se sont posés soit à cause de certains objectifs ambitieux, soit à cause de limitations tenant au langage utilisé.

1.3.6.1 Pour alléger le travail de l'utilisateur

Il s'agit de la non indépendance de certains MTC imposée par un enchaînement que l'on veut le plus automatique possible dans certains cas. Illustrons-le par un exemple : en mode exécution lors de l'interprétation d'une lecture sur l'idex du lecteur (en fait le fichier données courant) on s'aperçoit qu'il manque des données. La suite des opérations est alors : (en éliminant le cas où l'on concluerait à une erreur, ce qui semble trop peu souple) :

- attente commande après un message indiquant l'absence de données
- lancement d'une commande données
- lancement d'une commande de définition de fichier
- reprise de l'exécution.

(1) cf. Problème des adresses de retour dans les travaux de Monsieur DELPUECH.

Si l'on veut que la partition se substitue à l'utilisateur pour enchaîner ces opérations, il faut permettre qu'un MTC puisse en appeler un autre sans passer par le lancement d'une nouvelle commande. Cela veut dire qu'un MTC peut solliciter le moniteur local et plus précisément l'analyseur de commande.

Ceci est réalisé par une routine utilitaire qui appelle l'analyseur du moniteur local sur ce que nous appelons un pseudo-bloc. Ce pseudo-bloc est débuté par la partition qui demande à l'utilisateur de le compléter. (Ici, il apparaît xD et l'utilisateur complète la commande donnée comme si il l'avait entièrement émise).

Au moment de la définition éventuelle du fichier, il apparaît xDF qui est complété de manière analogue.

Ce procédé va à l'encontre du souci de modularité et d'indépendance et ne sera utilisé que dans des cas parfaitement justifiés. Cependant, cette solution est préférable à celle consistant à faire de l'opération DONNEES un SPTC car elle a le mérite de respecter les niveaux.

1.3.6.2 Un exemple de limitation dû au langage

Il s'agit de la non récursivité des sous-programmes Fortran. On ne peut pas réutiliser tout ou partie d'un MTC ou d'un SPTC par un appel pur et simple à partir de lui-même. Il faut faire de la séquence un sous-programme de niveau inférieur. Cette nécessité qui apparaissait au premier abord comme une limitation s'est révélée bénéfique. En effet, dans presque tous les cas il est apparu que de telles séquences relevaient en fait du niveau inférieur. Ce revirement d'opinion fait saisir à quel point il faut être attentif lors de la définition des niveaux. On mesure également l'intérêt du type d'empilement des machines qui permet de limiter l'impact d'une telle modification de niveau i au niveau i-1 seulement.

1.4 Position du traitement de commandes dans l'ensemble de la partition

La philosophie que nous venons de présenter conduit à considérer l'ensemble de tous les niveaux comme une machine globale chargée de réaliser "le traitement des commandes" dans son ensemble.

Nous venons de présenter sa structure interne et il nous reste à définir en amont les commandes et en aval les zones de travail de l'utilisateur.

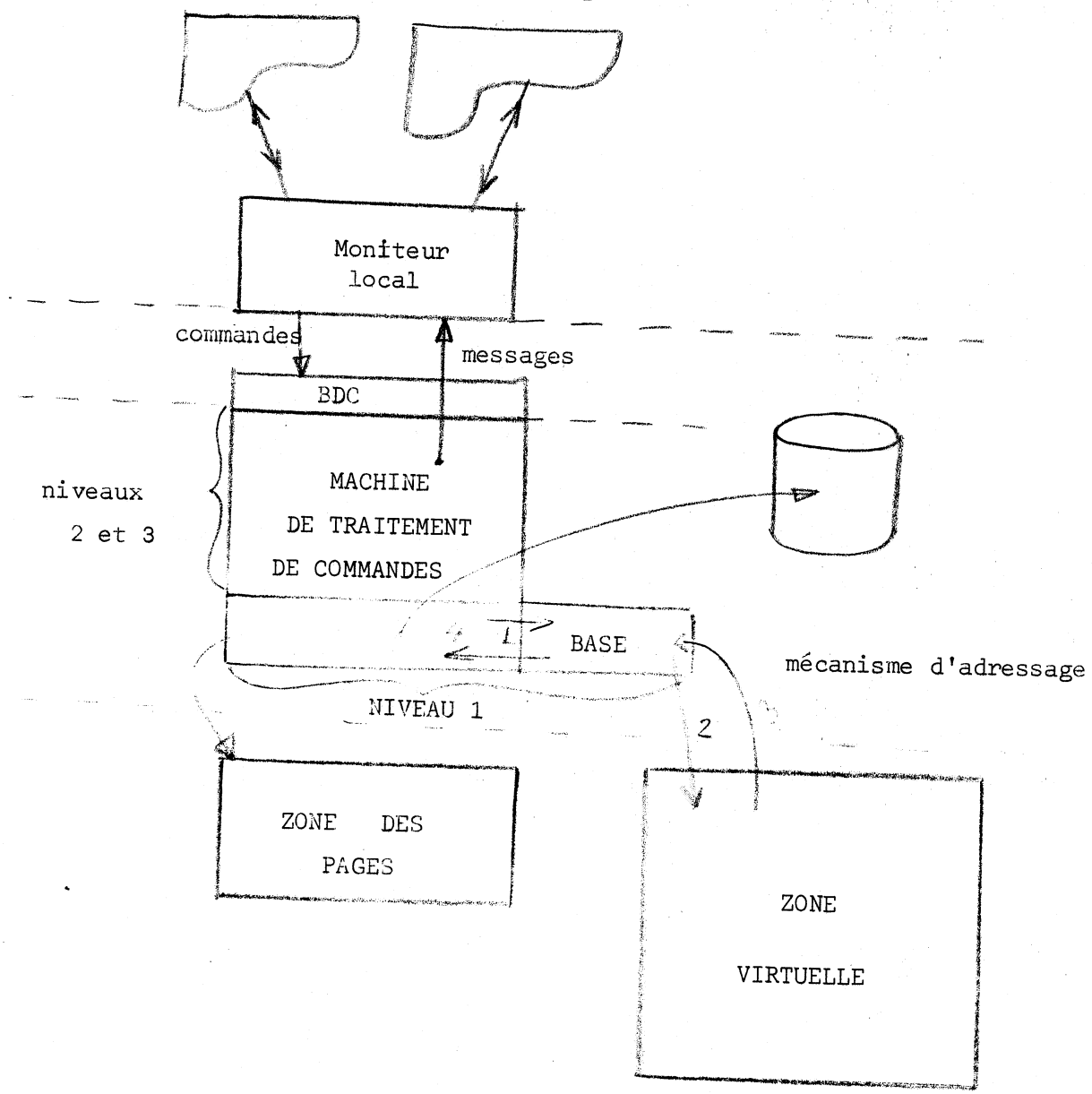


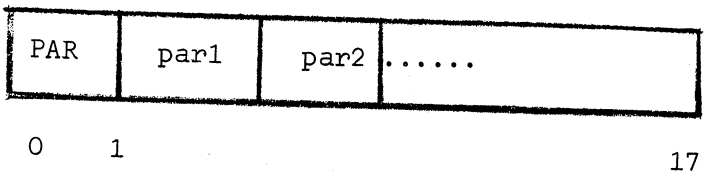
Schéma de la partition du point de vue Traitement de Commande

(Ce schéma n'est pas l'organisation interne du système. Il traduit seulement la philosophie de réalisation et considère le traitement des commandes comme un sous-système).

1.4.1 Prise en compte des commandes

Les commandes sont reçues par l'analyseur de commande du moniteur local qui réalise le branchement au MTC adéquat et lui fournit des informations sur le décodage de la commande dans le BDC (bloc de décodage des commandes) (1).

Le BDC a le format suivant :



par est un masque : si le $i^{ème}$ bit est à 0 le $i^{ème}$ paramètre est spécifié.
si il est à 1 c'est une valeur par défaut.

Chaque MTC connaît la signification et le format des paramètres et les mesures à prendre concernant les paramètres par défaut.

Chaque MTC commence par une initialisation éventuelle du BDC et une vérification de la correction des paramètres.

Remarque :

Il arrive qu'un MTC serve simplement de relais et se substitue au moniteur local pour réaliser le branchement à une autre MTC après quelques sauvegardes et initialisations.

C'est le cas en particulier des MTC opérant sur le fichier données courant au lieu du fichier source.

Ce mécanisme est décrit en appendice et montre une des applications de la page 0.

On peut également considérer que le niveau des MTC contient plusieurs machines qui peuvent communiquer entre elles.

(1) cf. Monsieur DELPUECH.

1.4.2 Les zones accessibles à l'utilisateur

On distingue les fichiers qui feront l'objet de la seconde partie et la zone de travail de l'utilisateur (ZT) dont nous allons parler maintenant.

Pour élaborer des programmes suffisamment importants pour un nombre raisonable d'utilisateurs, il fallait disposer d'une place mémoire telle qu'il était impensable de la rendre résidente. La place occupée était encore aggravée par le caractère incrémentiel du compilateur qui exigeait un grand nombre de tables.

La solution adoptée a été celle d'une mémoire virtuelle et d'un mécanisme de pagination software (1). Le format et l'organisation de la mémoire virtuelle reflète le désir d'éviter au maximum les dégradations de temps dues aux entrées/sorties de pagination.

Chaque utilisateur dispose d'une zone virtuelle de 128 K octets contenant toutes les informations nécessaires au cours d'une session. Cette zone est découpée en pages de taille 1/2 K octets (2).

D'un point de vue physique l'ensemble des mémoires virtuelles de n utilisateurs constitue un seul fichier disque géré en accés direct et composé de n 256 blocs de 1/2 K octets. Les n mémoires virtuelles sont donc contiguës et c'est un calcul d'adresse disque qui permet de déterminer la bonne page (3).

D'un point de vue logique, la mémoire virtuelle d'un utilisateur se présente comme un ensemble de tables et fichiers liés à des opérations spécifiques. (table d'identificateur, fichiers source et donnée courants, etc ...). A ces tables qui occupent une place de taille variables - mais multiples de 1/2K - il faut ajouter une page particulière,la page zéro. Nous allons présenter quelques éléments de cette mémoire virtuelle qui sont très importants dans l'application particulière.

(1) et (2) Voir les travaux de Monsieur DELPUECH et la thèse de Melle Joëlle COUTAZ concernant la simulation du système.

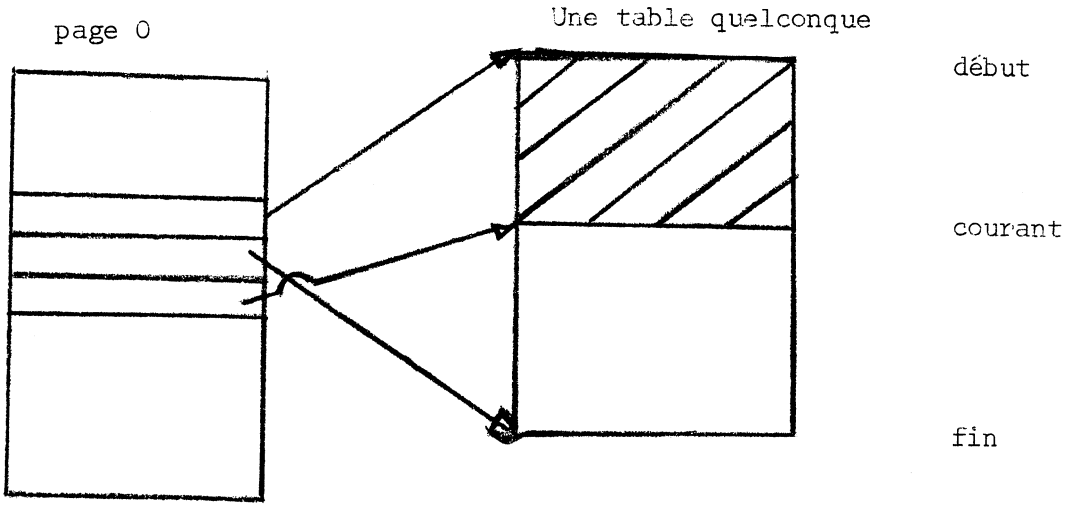
(3) on trouvera dans les appendices fichier une description de la lecture et de l'écriture d'une page.

1.4.2.1 La page 0

C'est une page qui contrôle toute la zone de travail de l'utilisateur. Elle contient tous les pointeurs sur le reste de la mémoire virtuelle (pointeurs virtuels), des informations de contrôle, les octets évènements, etc ... (1). Cette page permet à tout instant de connaître l'état de remplissage des différents fichiers et de prendre les dispositions d'enchaînement nécessaires. La page 0 est commune aux différents modes et sert également de moyen de communication minimum entre un MTC et un SPTC. La fréquence d'utilisation de cette page a conduit à la rendre résidente dès que l'utilisateur a initialisé sa session.

1.4.2.2 Les tables et les fichiers

Les différents éléments sont regroupés par mode pour des raisons d'initialisation. On y trouve également des zones qui sont concernées dans tous les modes. Nous parlerons des éléments les plus significatifs :



Contrôle d'une table à partir de la page 0

(1) Un appendice décrit en détail le format de certaines tables remarquables.

Le mécanisme d'accés de la table à partir d'une SPTC appelé par un MTC est le suivant :

- Le MTC met à jour les pointeurs.
- Il appelle le SPTC qui sait quels pointeurs utiliser et les déplacements en page 0.
- La page 0 donne le pointeur virtuel et un CALL BASIN1 permet d'avoir l'adresse réelle dans une base.

1.4.2.2.1 Les fichiers courants source et données

Il y a deux fichiers courants, un pour le source et un pour les données. Le format est variable, chaque entrée est constituée du texte précédé de la longueur sur deux octets, jusqu'à concurrence de la longueur maximum admise.

Le système de repérage de ces fichiers est décrit en appendice.

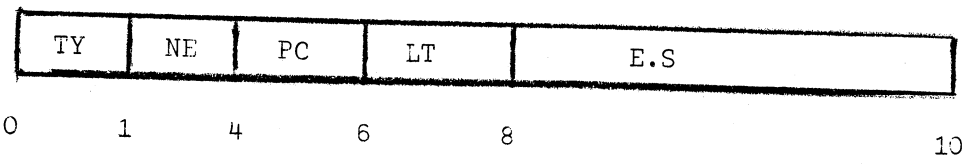
1.4.2.2.2. Le fichier courant objet

Il est identique aux précédents et contient le pseudo-code généré en raison d'une entrée par incrément.

1.4.2.2.3 La table des enchaînements (TE)

C'est un des éléments fondamentaux de la partition.

Le format d'une entrée de la table des enchaînements est le suivant :



Octet 0 : TY : type de ligne texte associé, c'est-à-dire donnée ou un des types possibles d'ordre Fortran.

Octets 1, 2, 3 : NE : numéro externe de la ligne texte composé de 2 octets pour la partie entière et 1 octet pour la partie décimale.

Octets 4, 5 : PC : pointeur sur le pseudo-code. (le pointeur n'est pas significatif pour les lignes données et les lignes texte erronées).

Octets 6, 7 : L.T. : pointeur sur le texte de la ligne dans le fichier texte : (ce champ a toujours une signification).

Octets 8, 9 : E.S. : pointeur sur l'élément suivant dans la table des enchaînements.

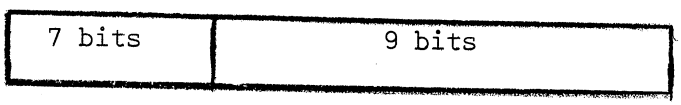
Chaque entrée occupe donc 10 octets et l'on admet 700 lignes texte, ce qui correspond à un encombrement de 7 K octets en mémoire virtuelle.

Remarque 1 :

Il y a une chaîne des éléments libres, une chaîne des éléments associés au FSC et une chaîne des éléments associés au FDC.

Remarque 2 :

PC, LT, ES sont des adresses virtuelles sur deux octets selon le format :



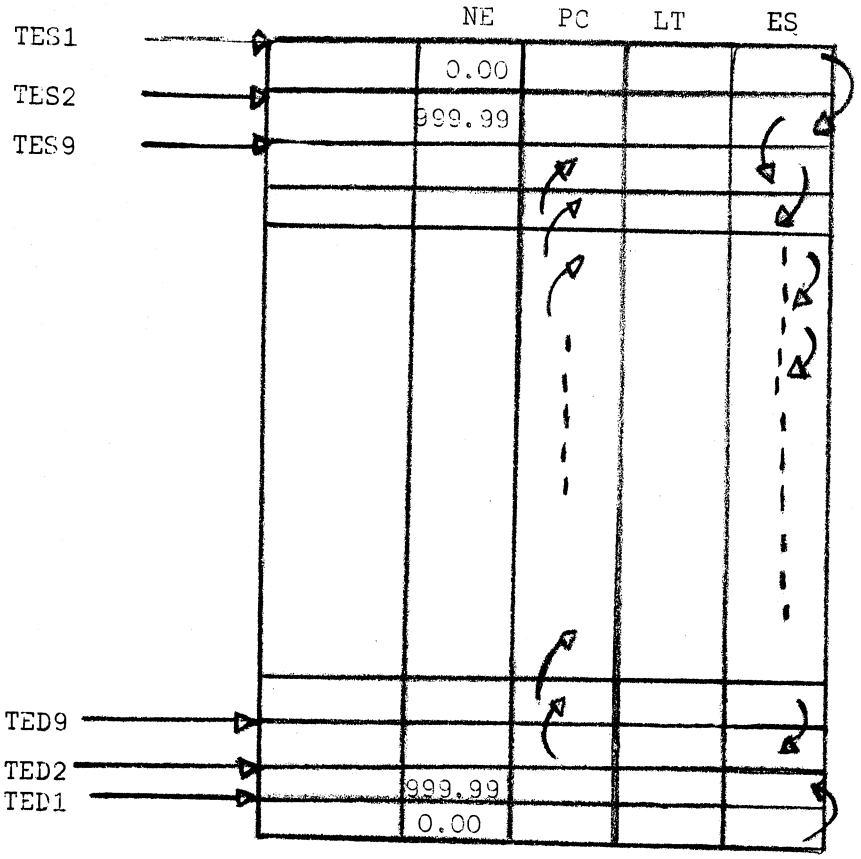
Ceci permet d'adresser les 64 K octets de mémoire virtuelle nécessaires.

Remarque 3 :

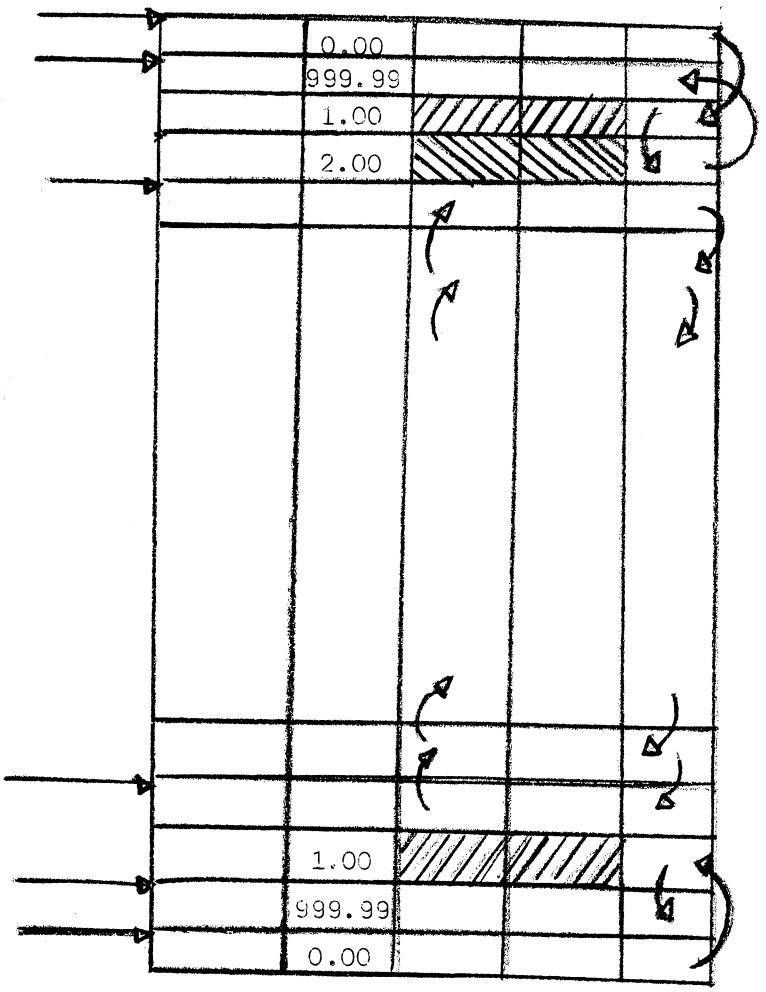
L'organisation des lignes texte dans la table des enchaînements est la suivante :

- Le FDC est rangé à partir de la fin.
- Le FSC " " " " du début.
- La chaîne libre est obtenue par un double chaînage, le chaînage avant étant réalisé pour le source par ES et le chaînage arrière pour les données par PC.
- Chaque fois que l'on insère un élément on modifie le double chaînage et l'on chaîne également soit au source soit aux données.
- Le premier élément a un numéro externe de 0.00
- Le dernier élément a un numéro externe de 999.99.

On peut représenter cette organisation par le schéma suivant :



Chaînage initial



Chaînage avec 2 lignes source et une donnée

Les chaînages en T.E.

1.4.2.2.4 Les autres tables : leur contenu ne sera pas décrit

On s'intéressera à l'emplacement des frontières entre les différentes tables qui n'est pas indifférent.

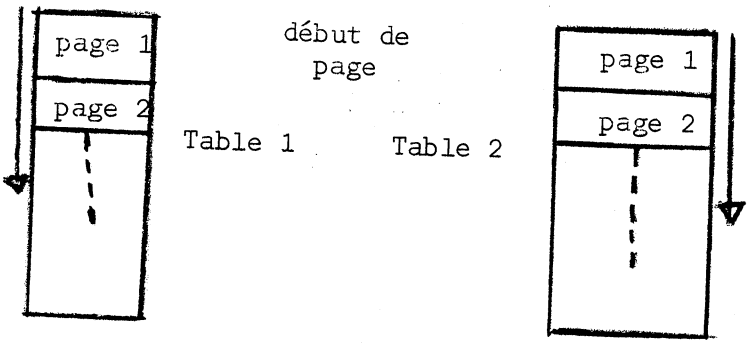
En effet, la pagination software est lente et il est important de minimiser les appels de pages. L'algorithme de pagination s'y emploie mais l'organisation des tables y contribue beaucoup.

La structure de la mémoire virtuelle est complice de la programmation.

Notre souci est de gréver en temps un utilisateur suivant la quantité de ressources qu'il demande.

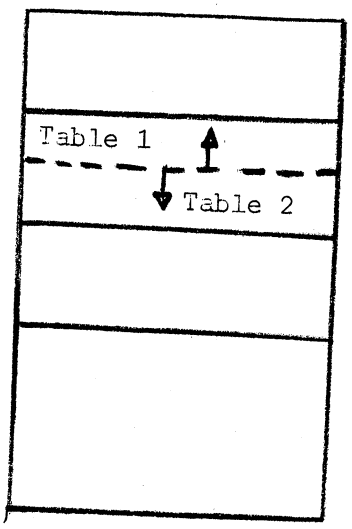
On désire en particulier satisfaire le plus rapidement possible des utilisateurs mettant en oeuvre de petits programmes. Ceci implique que soient présents en mémoire les tables nécessaires.

Si alors on met dans une même page deux débuts de tables différents mais utilisés simultanément, on gagne au moins un appel de page en début de travail.



Dans cette configuration un travail sur les 2 débuts de table demande 2 appels.

tandis que la disposition suivante



demande un seul appel

2 - UNE STRATEGIE DE REALISATION

2.1 Considérations générales

Jusqu'à présent on a implicitement supposé que tous les éléments décrits existeraient à un certain moment. Nous allons maintenant introduire le temps et indiquer dans quel ordre il semble intéressant de réaliser effectivement les différents éléments. On va définir un certain nombre d'étapes et indiquer pour chacune d'elles quelle partie des objectifs est couverte et quels sont les problèmes résolus. Ceci représente la stratégie de réalisation.

2.1.1 Les alternatives

Pour définir un ordre de réalisation, nous devons envisager les avantages et les inconvénients de plusieurs méthodes : "le haut en bas", et le "bas en haut", dont nous rappellerons les caractéristiques (1).

Le haut en bas :

C'est la méthode qui consiste à commencer par le niveau le plus élevé et à terminer par le niveau de base (ou primitives selon une terminologie de DIJKSTRA). Cette méthode particulièrement recommandée quand les objectifs du système sont parfaitement précisés sans que la machine destinée à le recevoir le soit.

Le bas en haut :

Cette méthode est très appropriée dans le cas où le hardware et les primitives sont bien définies alors que le système l'est seulement dans ses grandes lignes.

(1) cf. Bibliographie.

Il semble donc que la stratégie à adopter dépende de l'état de précision des définitions de la machine d'une part, du système d'autre part.

On doit en outre remarquer qu'il est tout à fait envisageable et même souhaitable de changer de tactique lorsque l'on passe d'une phase du travail à l'autre. En effet, dans la phase de définition générale du système qui vise à la plus grande généralité possible on aura tendance à ignorer la machine. Ensuite, lors de l'étude de faisabilité, les objectifs étant définis, on se préoccupera de la machine sur laquelle le système doit fonctionner.

Voyons maintenant les problèmes posés par chacune de ces méthodes. Il semble que dans les deux cas les difficultés interviennent aux niveaux intermédiaires du fait de la propagation de certaines caractéristiques à travers les niveaux. Par exemple, dans la méthode de "haut en bas" on peut rencontrer au dernier niveau de très grandes difficultés pour résoudre des problèmes tenant aux objectifs du système et qui auraient dû être résolus en amont. Dans la méthode de "bas en haut" au contraire certaines particularités du hardware ont tendance à se répercuter dans les niveaux supérieurs alors qu'il aurait été préférable de les faire disparaître (dans les langages évolués par exemple).

La valeur de l'une ou l'autre approche dépend de la capacité du concepteur à anticiper de tels problèmes et à y trouver des solutions.

En effet, à un niveau donner la décision de doter la machine d'une certaine fonction a des incidences en amont : il y a la fonction proprement dite mais aussi les possibilités qu'elle donne au niveau supérieur. Les définitions concernent donc à chaque étape deux niveaux simultanément. Lorsqu'un critère a hiérarchisé les niveaux, il faut définir les fonctions nécessaires dans un niveau mais également s'assurer qu'elles sont suffisantes pour que le niveau supérieur soit celui souhaité.

C'est dans l'étude de l'incidence en amont que le concepteur doit faire preuve de la plus grande intuition. Cette remarque permet en partie de choisir la tactique : dans la méthode de "bas en haut" la définition d'une fonction doit être faite sans ambiguïté et complètement si l'on ne veut pas revenir en arrière. Dans le procédé de haut en bas il suffit de les définir quant à leur rôle dans l'édifice total en travaillant avec un niveau d'avance. La définition et la programmation complétes peuvent être reportées à plus tard.

Dans la tactique de "bas en haut", on travaille donc sur des composants imparfaitement définis. Il faut alors être particulièrement prudents et s'assurer qu'en complètant la définition on n'introduira pas d'effets de bord nuisibles qui n'avaient pas été prévus dans les niveaux ou on avait eu recours à cette définition.

Connaissant les motifs et les mouvements de ces deux méthodes, voyons quelle stratégie de réalisations a été adoptée et pour quels motifs.

2.1.2 Le choix

On se place pour tout ce qui suit dans la phase des spécifications de réalisation c'est-à-dire à un stade où les objectifs généraux du système sont définis avec une grande précision. C'est, en effet, la phase où le choix d'une méthode est le plus ardu.

Nous avons adopté un compromis qui tient compte des considérations générales ci-dessus et du désir d'élaborer un produit facilement modifiable, extensible, et dont la plus grande partie soit indépendante de la machine. C'est-à-dire que l'on envisageait une modification possible des objectifs d'une part et un fonctionnement sur une autre machine (1) d'autre part.

Il est alors apparu un niveau intermédiaire à partir duquel les stratégies allaient changer. Ce niveau est défini comme suit : Au-dessus, le changement en hardware n'a pas d'incidence; au-dessous, on fournit un ensemble de fonctions à caractère général qui sont nécessaires et suffisantes quelque soient les objectifs du système. Il est bien évident que ce niveau est celui des routines utilitaires. De part et d'autre de cette coupure, les stratégies seront différentes : en dessus, on fera "du haut en bas", en dessous, "du bas en haut" puisque pour l'instant on a un seul objectif et une seule machine.

On peut déjà voir quelle importance revêt le niveau intermédiaire et avec quel soin il faudra choisir le langage de programmation. Une bonne définition de ce niveau donne une grande souplesse au produit. En effet, un changement d'objectif se traduira par une inversion de la stratégie de réalisation au-dessus, et un changement de hardware par une inversion en dessous.

(1) Cette remarque a permis la simulation du niveau physique à Grenoble.

2.2 Les trois phases

Nous venons de voir que la stratégie choisie s'applique surtout à partir de la phase des spécifications. Cependant, la phase préliminaire doit préparer l'application de cette stratégie en prenant en considération la répartition du travail compte tenu des outils de réalisation et de tests.

2.2.1 La phase de définition générale

Au cours de cette phase, les objectifs généraux du système ont été définis. En particulier, les différentes commandes ont été définies. Nous allons les présenter et les décrire succintement afin de préciser les idées sur la nature et la taille du produit.

2.2.1.1 Commandes d'élaboration du fichier courant

(Mode associé : compilation).

Le fichier courant s'élabore dans la ZT allouée à l'utilisateur. Il se décompose en :

- un fichier source courant (FSC)
- un fichier données courant (FDC)
- un fichier objet (FO)

2.2.1.1.1 Présentation des commandes

Les commandes d'élaboration du fichier courant se répartissent en deux groupes principaux :

Commandes relatives au fichier source courant et/ou au fichier objet :

COMPILER qui permet la compilation d'une ou plusieurs nouvelles lignes.

MODIFIER qui permet l'appeler une ligne du fichier source que l'on désire modifier.

INSERER qui permet l'insertion d'une séquence de lignes du FSC entre deux lignes du même fichier.

EFFACER qui permet la suppression de certaines lignes à la fois dans le FSC et dans le FO.

LISTER qui permet de lister tout ou partie du FSC.

NUMEROTER qui permet de redonner à chaque ligne source un numéro externe.

LISTER NUMEROTER qui combine les effets des deux commandes précédentes.

Commandes relatives au fichier données courant

Appartiennent à ce groupe les commandes présentées précédemment mais spécialisées au traitement de lignes données :

DONNEES

MODIFIER DONNEES

INSERER DONNEES

EFFACER DONNEES

LISTER DONNEES

NUMEROTER DONNEES

LISTER NUMEROTER DONNES

2.2.1.1.2 Description des commandes

A) COMPILER

C<n>, <p>, <NOM>, L

1°) Valeur des paramètres

Paramètre	Valeur du paramètre spécifié	Valeur du paramètre par défaut
<n>	Numéro externe d'instruction	Numéro externe supérieur, d'une valeur égale au pas spécifié ou par défaut, au numéro externe de la dernière instruction du FSC Si le FSC est encore vide <n> = 1
<p>	Pas, de la forme nn.nn	p = 1, .1, .01 suivant que <n> spécifié comporte 0, 1 ou 2 décimales. Si <n> est par défaut c'est l'ancien pas qui est pris (s'il n'y en avait pas, alors <p> = 1).
<NOM>	Nom de fichier source	"Le fichier source dont les instructions vont être tapées au TI"
L	"avec liste imprimée au TI du fichier source <NOM>"	"sans liste imprimée au TI du fichier source NOM ".

2°) Action et bloc engendré au TI lorsque <NOM> n'est pas spécifié

Si <NOM> n'est pas spécifié, le FSC (et le FO) vont être constitués à partir d'instructions tapées au TI.

Dès que la PC est prête à compiler la première instruction, elle lance un IL qui est le numéro externe (spécifié ou par défaut) <n> contenu dans la ligne commande.

L'utilisateur peut alors taper la première instruction qui se termine par un RC .

L'instruction est compilée.

La compilation terminée, la PC lance un deuxième IL numéro externe d'instruction déduit du premier par l'addition du pas (spécifié ou par défaut).

L'opération peut recommencer de la même manière jusqu'à ce que l'utilisateur tape un RC isolé en réponse à un numéro externe. Un nouveau bloc est alors initialisé par l'IL **.

S'il arrive, en cours d'élaboration du FSC, que, par l'addition du pas au numéro externe précédent, la PC trouve un numéro externe qui existe déjà dans le FSC, il y a erreur, et fin anormale du bloc.

Exemple :

```
** C 10,2.5
 10    :1 I=5
 12.5  : A=I
 15    :WRITE (6,5) I
 17.5  :5 FORMAT (I20)
 20    :RC
**
```

Commentaires

Ces instructions seront rangées dans le FSC après l'instruction dont le numéro externe est le plus voisin de 10 par valeurs inférieures. Ainsi ces 4 instructions peuvent aussi bien être insérées dans le fichier source (si par exemple il existe des numéros externes tels que 9 et 11) que concaténées à la fin du FSC (si par exemple le plus grand numéro externe est 8).

Il est à remarquer que, bien que le numéro externe 20 ait été utilisé une fois, il ne correspond à aucune instruction du FSC, et qu'ainsi il pourra être réutilisé ultérieurement.

Problème de la normalisation des instructions

Dès que la frappe d'une instruction Fortran (comprenant une ou plusieurs lignes) est terminée, l'instruction est normalisée sous la forme d'une suite de lignes de 63 caractères de long, où la première est telle que :

- l'étiquette est cadrée à droite dans les positions 1 à 5
- la position 6 est blanche
- l'instruction est cadrée à gauche dans les colonnes 7 à 63.

et où les suivantes sont telles que :

- les colonnes 1 à 5 sont blanches
- la colonne 6 contient à caractère de prolongation
- la suite de l'instruction est cadrée à gauche dans les colonnes 7 à 63.

De plus, tous les signes de prolongation tapés en fin de ligne au TI par l'utilisateur ont été remplacés par des blancs.

A partir de ce moment-là, c'est la forme normalisée qui, seule sera utilisable quelle que soit l'opération à effectuer (liste, modification, sauvegarde).

Normalisation de l'exemple précédent

```
  17 : 6 FORMAT(17HCECI EST UN ESSAI,
     :  118X,  3H421    ,14X
     :  1I8,(E12.5))
  18 :
```

Problème des erreurs décelées à la compilation

Dans le cas où une ou plusieurs erreurs sont décelées lors de la compilation de la dernière instruction entrée au TI, l'entrée des lignes source est momentanément interrompue et tout est fait pour que l'utilisateur puisse corriger immédiatement la ou les erreurs.

Pour cela

- le ou les messages d'erreur sont imprimés au TI.
- l'instruction erronée est alors listée (elle est sous forme normalisée).
- l'utilisateur est sollicité pour effectuer une correction au moyen de l'IL MODIF :
- la ligne corrigée est alors listée.
- l'utilisateur est à nouveau sollicité pour faire une correction et la même séquence se reproduit tant que l'utilisateur fait des corrections.
- lorsque l'utilisateur a fini ses corrections il tape RC après l'IL MODIF :

- la ligne est alors à nouveau compilée.
- s'il y a une erreur la situation est la même que lors de l'impression du premier message d'erreur.
- s'il n'y a pas d'erreur l'entrée des lignes source suivantes continue par l'envoi d'un IL numéro externe de ligne.

Exemple :

```
 17   :
-C-409:ERREUR:MANQUE VALEURFINALEVAR.CONTROLEE
 17   :    4 DO8 I=1
 MODIF:|,10|
 17   :    4 DO8 I=1,10
 MODIF:RC
 18   :
```

Remarque :

Si l'utilisateur tape uniquement RC lors du premier IL MODIF :, la ligne n'est pas recompilée et l'entrée des lignes source continue immédiatement.

3°) Action et bloc engendré au TI lorsque NOM est spécifié (1)

Si <NOM> est spécifié le FSC (et le FO) vont être constitués à partir d'un fichier référencé par <NOM> .

Ce fichier peut avoir 4 origines (1) :

- la BPC de l'utilisateur
- la BCC
- une BPB
- un ruban perforé

(1) cf. Seconde partie sur les fichiers.

Dans les 3 premiers cas, si ce fichier a été créé avec des renseignements non encore fournis par l'utilisateur, ceux-ci lui seront réclamés.

Ensuite, la compilation se fera comme dans le cas de lignes provenant du TI.

Cependant, suivant que l'utilisateur aura spécifié ou non le quatrième paramètre L, la liste avec les numéros externes apparaîtra ou non sur le TI.

Problème de la normalisation des instructions

De toute évidence, les lignes en provenance de tels fichiers sont normalisées : aussi bien en ce qui concerne le format de la ligne que les caractères de prolongation.

B) MODIFIER

M <n>

Paramètre	Valeur du paramètre spécifié
<n>	Numéro externe d'instruction du FSC

Ce paramètre est obligatoire

1°) Action et bloc engendré au TI (cf. exemple)

L'instruction du FSC dont le numéro externe est celui spécifié par <n>, est listée au TI.

L'utilisateur peut ensuite indiquer la correction à effectuer en répondant à l'IL MODIF :

L'instruction corrigée est alors listée.

L'utilisateur peut indiquer une autre correction pour la même instruction car il est sollicité par un autre MODIF :

Cette liste de corrections ne s'arrête que lorsque l'utilisateur tape un RC isolé après un MODIF :

L'instruction corrigée est alors compilée. De ce fait, il est possible que des messages d'erreur soient imprimés au TI à la suite de l'instruction modifiée.

Lorsque la liste de corrections est arrêtée et que l'instruction est correcte pour le compilateur, un nouveau bloc est initialisé par l'IL **.

A la sortie du bloc MODIFIER, l'instruction du FSC référencée par le numéro externe <n> est celle qui a été listée en dernier au TI.

2°) Spécifications des corrections à effectuer

Echange de caractères (cf. Exemple 1)

Pour remplacer dans l'instruction à corriger, la première occurrence de la chaîne de caractères <CH1> par la chaîne de caractères <CH2>, l'utilisateur tape

MODIF : <CH1>|<CH2>|

<CH1> et <CH2> sont des chaînes de longueur quelconque. Il faut simplement que ces deux chaînes puissent rentrer dans la même ligne tapée au TI (il ne peut y avoir de caractère de prolongation dans une ligne commençant par MODIF :)

Concaténation de caractères (cf. Exemple 2)

Si <CH1> a une longueur nulle, la convention adoptée est que la <CH2> est concaténée à l'instruction à corriger.

Suppression de caractères (cf. Exemple 3)

Si <CH2>a une longueur nulle, la première occurrence de <CH1> est supprimée de la ligne.

Restauration de l'ancienne ligne (cf. Exemple 5)

Si l'utilisateur tape '&' en réponse à MODIF :, toutes les modifications apportées à l'instruction Fortran dans le cadre du bloc courant associé à la commande MODIFIER, sont annulées et l'instruction listée au TI est

identique à celle qui avait été listée avant la première correction.

3°) Remarques pratiques concernant les corrections

1 - Dans <CH1> et <CH2>, le blanc est significatif.

2 - Que l'instruction Fortran fasse une ou plusieurs lignes, lorsque l'utilisateur la corrige, il ne peut en fait corriger qu'une ligne au cours d'une correction.
De ce fait lorsque la longueur de <CH2> est plus grande que la longueur de <CH1> tous les caractères ayant dépassés la 63ème position dans la ligne corrigée sont perdus. (et non pas reportés en début de la ligne suivante).(Inversement des blancs apparaissent en bout de ligne et non les caractères de la ligne suivante).

3 - Après chaque correction, la ligne modifiée est à nouveau normalisée.

4°) Exemple

```
          ** M 53.1
           53.1 : 1000 X=A+B*I
Ex1        MODIF:A|ARC| RC
           53.1 : 1000 X=ARC+B*1
Ex2        MODIF:|+C| RC
           53.1 : 1000 X=ARC+B*I+C
Ex3        MODIF:B*I+| RC
           53.1 : 1000 X=ARC+C
Ex4        MODIF:1000|| RC
           53.1 :      X=ARC+C
Ex5        MODIF:& RC
           53.1 : 100 X=A+B*I
           MODIF: RC
          **
```

C) INSERER

I<SI>, <n>, <p>, L

Paramètre	Valeur du paramètre spécifié	Valeur du paramètre par défaut
<SI>	Séquence d'instructions du FSC	Ce paramètre est obligatoire
<n>	Numéro externe d'instruction	Ce paramètre est obligatoire
<p>	Pas, de la forme nn.nn	<p> = 1 , .1 ou .01 si <n> comporte 0, 1 ou 2 décimales
L	"avec liste au TI de la séquence <SI>"	"sans liste au TI de la séquence <SI>".

1°) Action

La séquence d'instructions du FSC spécifiée par <SI> est insérée après l'instruction du FSC dont le numéro externe est le plus proche de n par valeurs inférieures.

Les instructions de la séquence <SI> sont renumérotées à partir de <n> par pas de <p>.

2°) Bloc engendré au TI

Si L est spécifié la liste des instructions apparaît au TI avec les nouveaux numéros externes.

Dès que l'insertion est terminée, un nouveau bloc est initialisé par un IL **.

3°) Remarques pratiques

1 - La valeur de <n> ne doit pas être comprise entre les numéros externes de la première et de la dernière instruction de la séquence <SI>.

2 - La valeur de <n> ne doit pas exister déjà dans le FSC.

3 - Les instructions de la séquence <SI> ne conservent pas leur ancien numéro externe et elles ne pourront être désormais référencées que par les nouvaaux numéros.

4 - La commande INSERER correspond à un déplacement d'instructions déjà existantes et non à une duplication.
C'est donc que les instructions de <SI> ne se trouvent plus à leur ancienne place.

5 - Pour dupliquer une séquence assez importante d'instructions du FSC, l'utilisateur aura intérêt à la sauvegarder dans sa BPC ou sur ruban puis à la compiler à partir de là et la placer ainsi à l'endroit qui lui convient dans le FSC.

D) EFFACER

EF<SI>

Paramètre	Valeur du paramètre spécifié
<SI>	Séquence d'instructions du FSC

Ce paramètre est obligatoire

Action et bloc engendré au TI

A moins d'une erreur dans la ligne commande, seule la ligne commande apparaît dans le bloc.

Lorsque la séquence d'instructions spécifiée par <SI> est effacée du FSC et FO, un nouveau bloc est initialisé avec un IL **.

E) LISTER

L <SI>, P

Paramètre	Valeur du paramètre spécifié	Valeur du paramètre par défaut
<SI>	Séquence d'instructions du FSC	L'ensemble du FSC
P	"avec présentation particulière"	"sans présentation particulière"

1°) Action et bloc engendré dans le cas où P n'est pas spécifié

La séquence d'instructions du FSC spécifiée par <SI> est listée avec les numéros externes correspondants.

Lorsque la liste demandée est toute imprimée, un nouveau bloc est initialisé.

2°) Action et bloc engendré dans le cas où P est spécifié

Un avancement du papier est réalisé au TI.

Certaines informations dont la date et le nom de l'utilisateur sont imprimées.

La séquence d'instructions du FSC est listée sans numéros externes. Lorsque la liste demandée est terminée, un avancement du papier est réalisé et un nouveau bloc est initialisé.

F) NUMEROTER

N<n1>, <n2>, <p>

Paramètre	Valeur du paramètre spécifié	Valeur du paramètre par défaut
<n1>	Numéro externe d'instruction du FSC	Numéro externe de la première instruction du FSC
<n2>	Numéro externe d'instruction	<n1>
<p>	Pas, de la forme nn.nn	<p> = 1, .1, .01 suivant que <n2> comporte 0, 1 ou 2 décimales

1°) Action

L'instruction du FSC dont le numéro externe est <n1> prend le numéro <n2>.

Ensuite toutes les lignes suivantes du FSC sont renumérotées par pas de <p> à partir de <n2>.

2°) Bloc engendré au TI

La seule ligne du bloc est la ligne commande.
Lorsque la numérotation est terminée un nouveau bloc est initialisé.

3°) Remarques pratiques

1 - Le FSC est renuméroté depuis l'instruction <n1> jusqu'à la fin.

2 - n2 doit être supérieur ou égal à <n1>.
Ceci afin qu'il n'y ait pas dans le FSC deux instructions ayant même numéro externe.

3 - Si aucun paramètre n'est spécifié, le FSC est entièrement numéroté depuis 1 par pas de 1.

Commandes relatives au fichier données courant

DONNEES	D<n>, <p>, <NOM>, L
MODIFIER DONNEES	MD<n>
INSERER DONNEES	ID<SI>, <n>, <p>, L
EFFACER DONNEES	ED<SI>
LISTER DONNEES	LD<SI>,<P>
NUMEROTER DONNEES	ND<n1>, <n2>, <p>
LISTER NUMEROTER DONNEES	LND<SI>, <n2>, <p>

Toutes ces commandes, permettant la manipulation de données, ont leur homologue permettant de travailler sur des instructions Fortran. Les conventions sont les mêmes aux différences près suivantes :

1 - Les lignes données remplacent les instructions Fortran.
2 - Une ligne données a une longueur maximale de 63 caractères.
3 - Il n'y a ni compilation, ni normalisation faites des lignes données.
4 - Il n'y a aucune limitation sur les caractères admis en données.
5 - Il ne peut y avoir de signe de prolongation.

G) TOTAL

TO

1°) Bloc engendré au TI

Un message lancé au TI par la PC permet à l'utilisateur de savoir combien il y a de lignes dans son FSC ainsi que dans son FDC et également, de savoir combien il lui reste de caractères disponibles pour ranger des lignes sources et/ou de données.

2°) Remarques pratiques

L'utilisateur, connaissant la place qu'il lui reste sait s'il doit faire usage de la commande NETTOYER.

3°) Exemple :

```
** TOTAL
-C-900:FSC=213 FDC=114 LIGNES 327/768 CARACT. 12403/32768
**
```

Commentaire

Il y a 213 instructions dans le FSC.
Il y a 114 lignes dans le FDC.
Il y a 327 lignes au total (le maximum est de 768).
Ces 327 lignes données et instructions Fortran occupent 12403 caractères en ZT (le maximum est 32768).

2.2.1.2 Execution du fichier courant

(mode associé : exécution)

2.2.1.2.1 Présentation des commandes

Les commandes d'exécution se répartissent en trois groupes :

- Les commandes qui permettent de simuler, avec des éléments symboliques tels que numéro externe de ligne ou nom de variable ou de tableau, toutes les fonctions pupitre.

EXECUTER qui permet d'exécuter tout ou partie du fichier source courant en utilisant tout ou partie du fichier données courant

PAS A PAS qui permet l'exécution du FSC, instruction par instruction

REPARTIR qui permet de relancer l'exécution du FSC après la dernière instruction exécutée avant un BREAK

RANGER qui permet d'affecter des valeurs à des variables ou à des tableaux

IMPRIMER qui permet d'imprimer au TI les valeurs de certaines variables et/ou de certains tableaux.

- Les commandes concernant la sortie des résultats au TI

PERFORER RESULTATS
IMPRIMER RESULTATS
PERFORER IMPRIMER RESULTATS

qui permettent de perforer et/ou d'imprimer les résultats du programme courant

- Les commandes qui sont des aides à la programmation

TRACER qui permet l'impression dynamique (c'est-à-dire sans arrêt de l'exécution) des valeurs de certaines variables et/ou de certains tableaux en certains points particuliers du FSC.

DESACTIVER TRACER qui permet d'arrêter tout ou partie de la trace précédemment lancée.

ACTIVER TRACE qui permet d'arrêter tout ou partie de la trace précédemment arrêtée.

2.2.1.2.2 Description des commandes

A) EXECUTER

E<SI>, <SL>

Paramètre	Valeur du paramètre spécifié	Valeur du paramètre par défaut
<SI>	Séquence d'instructions du FSC commençant et finissant respectivement aux instructions de numéro externe <n1> et <n2>	L'ensemble du FSC
<SL>	Séquence de lignes du FDC commençant et finissant respectivement aux lignes de numéro externe <n3> et <n4>	L'ensemble du FDC

1°) Action

L'exécution du FSC commence à l'instruction de numéro externe <n1>, la première ligne du FDC à lire est celle de numéro <n3>.

L'exécution du FSC sera arrêtée de façon normale dès que l'un des deux évènements suivants arrivera :

- l'instruction de numéro <n2> a été exécutée
- le programme a essayé de lire dans le FDC au-delà de la ligne <n4>

L'exécution pourra être arrêtée de façon normale de 2 façons possibles

- par un BREAK de l'utilisateur
- par une erreur détectée par l'interpréteur dans une instruction du FSC

2°) Bloc engendré au TI (cf. Exemple)

Après que l'exécution ait été lancée, d'éventuelles lignes texte de différents types peuvent apparaître au TI :

- il peut d'abord y avoir des lignes résultant d'instruction Fortran WRITE. Ces lignes apparaissent sans IL (cf. Exemple 1).

- ensuite la PC peut réclamer des informations supplémentaires à l'utilisateur et ceci pour deux raisons possibles :

  1 - L'interpréteur a rencontré une instruction Fortran qui utilise un fichier non défini.

  2 - L'interpréteur a rencontré une instruction READ qui doit lire une ou plusieurs lignes données dans le fichier données courant qui ne contient plus de lignes non lues en séquence. Il y a alors activation de la procédure "manque données".

Procédure "manque données"

- Elle commence par un message précisant à l'utilisateur ses "coordonnées" dans le fichier courant : c'est-à-dire d'une part le numéro externe de la dernière instruction exécutée et d'autre part le numéro externe de la dernière ligne de données lue (cf. Exemple 6).

- Ensuite un message indique à l'utilisateur qu'il n'y a pas assez de données dans son FDC (cf. Exemple 7).

- L'utilisateur est alors sollicité par l'IL *D qui initialise un pseudo-bloc DONNEES (il s'agit d'un pseudo bloc car en fait l'utilisateur n'est pas sorti du bloc associé à la commande EXECUTER (ou REPARTIR comme dans l'exemple). L'utilisateur devra donc commencer par spécifier les paramètres et suivant que <NOM> aura

  <n>, <p>,<NOM>, L de DONNEES (cf. Exemple 8).

été spécifié ou non, les lignes devant constituer la suite du fichier données courant seront lues dans le fichier <NOM> ou au TI conformément aux spécifications de la commande DONNEES

- Lorsque toutes les lignes données dournies auront été rangées dans le FDC, l'utilisateur sera averti par un message que l'exécution est repartie.

Cas où l'utilisateur tente une nouvelle fois de lire au-delà de la dernière ligne dans son FDC

1 - Si, la fois précédente, le fichier données courant a été complété à partir d'une bibliothèque ou d'un ruban,la procédure "manque données" est alors réactivée.

2 - Si,la fois précédente, le fichier données courant a été complété à partir du TI, la procédure activée est simplifiée :

- il sort d'abord un message indiquant les "coordonnées".

- puis un IL numéro externe de ligne données est tapé : une série de lignes données entrées au TI est ainsi initialisé

- l'utilisateur pourra l'arrêter par un RC isolé.

Cependant on peut imaginer que l'utilisateur, se retrouvant à nouveau sans ligne données disponible dans son FDC et ayant rentré la fois précédente des lignes à partir du TI, veuille utiliser un fichier données sauvegardée dans une bibliothèque ou sur ruban.

Si c'est le cas, dès le premier numéro externe qui est lancé, l'utilisateur tape un RC isolé.

Ceci a pour effet d'initialiser un pseudo-bloc DONNEES avec *D. La suite est conforme à la procédure "manque données".

Cas de fin normale du bloc **EXECUTER**

Si le bloc n'est constitué que de lignes résultats, de pseudo-blocs DEFINIR FICHIER et de pseudo-blocs DONNEES, et si l'exécution du fichier courant va jusqu'à l'instruction <n2> dans le FSC ou jusqu'à la ligne données <n4> dans le FDC, le bloc se termine normalement et un autre est initialisé par **.

Cas de fin anormale du bloc EXECUTER

Si une erreur est détectée par l'interpréteur, un message d'erreur, puis un message donnant les "coordonnées" sont lancés au TI et un nouveau bloc est initialisé.

Si l'utilisateur tape BREAK un message donnant les coordonnées puis un message de BREAK sont lancés et un bloc est également initialisé. Cependant, il est à remarquer que si un BREAK a été lancé en cours de procédure "manque données", l'instruction READ en cours n'aura pas été complètement exécutée et une nouvelle instruction d'exécution pourrait avoir une suite imprévue !

Exemple :

Supposons que le fichier courant de l'utilisateur comprenne :

- un fichier source de 100 instructions numérotée de 1 à 100
- un fichier données de 5 instructions numérotée de 201 à 205.

L'utilisateur possède de plus un ruban perforé de données de 50 lignes et un fichier données sauvegardé dans sa BPC sous le nom *DATA de 25 lignes (ce fichier est supposé défini sans mot de passe).

```
      ** E  /90,201/270
Ex1         X=3.15 Y=1.53                          ] lignes résultats
Ex2   -E-001:INST.:53    :DON.:204
Ex3   -E-206:ERREUR:54    :TABLEAU T NON DECLARE
Ex4   ** C 3.1
       3.1:DIMENSION T(10)
       3.2: (RC)
Ex5   ** R
          X=10.06  Y=1.53                         ] lignes résultats
Ex6   -E-001:INST.:76    : DON.:205
Ex7   -E-012:MANQUE DONNEES
Ex8    * D 206,,%,L
      206    : 9 6 3 2
      207    : 2 3 6 9                            ] liste des 50 lignes données du ruban
Ex10  -E-005:EXECUTION REPARTIE                    ] lignes résultats

Ex11  -E-001:INST.: 76    :DON.:255
      -E-012:MANQUE DONNEES
       * D 256
Ex12  256    : 1 2 3 4 5                           ]
      257    : 6 7 8 9 0                           ] données entrées au TI.
      258    :  (RC)                               ]
                                                   ] lignes résultats
Ex13  -E-001:INST.: 76    :DON.:257
Ex14  258    : 0 9 8 7 6
Ex15  259    :  (RC)                               ] données entrées au TI
                                                   ] lignes résultats
      -E-001:INST./ 76    :DON : 258
Ex16  259    :  (RC)
       * D 259,,*DATA
      -E-005:EXECUTION REPARTIE

Ex18  -E-001:INST.:50     :DON : 270               ] lignes résultats
      **
```

B) PAS A PAS

PAP <SI> , <SL>

Les paramètres ont la même signification que ceux de EXECUTER.

1°) Action et bloc engendré au TI

Cette commande est semblable à la commande EXECUTER à quelque différence près :

- l'exécution se fait instruction après instruction.
- Après l'exécution d'une instruction, la PC envoie un message indiquant le numéro externe de l'instruction qui vient d'être exécutée, ainsi que le numéro externe de la dernière ligne donnée qui a été lue.

Ce message n'apparaît qu'après l'impression au TI de toutes les lignes texte concernant l'instruction.

- L'utilisateur est ensuite sollicité pour demander l'exécution de l'instruction suivante.

Pour cela il répond n'importe quelle chaîne de caractères suivie d'un RC ou tout simplement un RC isolé à l'IL SUITE :.

2°) Remarque

Tout ce qui peut arriver dans le cadre d'un bloc EXECUTER peut arriver avec un bloc PAS A PAS : qu'il s'agisse de lignes résultats, de pseudo-blocs DEFINIR FICHIER, de pseudo-blocs DONNEES, des erreurs décelées par l'interpréteur ou d'un BREAK tapé en cours d'exécution.

EXEMPLE

```
        **PAP 1/5,10/30
(1)     -E-001:INST.: 1  :DON.:*
        SUITE:RC
        -E-001:INST.: 2  :DON.: 20
        SUITE:ALLONS-Y RC
```

] lignes résultats

```
-E-001:INST.:  3     : DON.:20
 SUITE: RC
-E-005:MANQUE DONNEES
 * D 21
21     : 1.2  3.4
22     :  RC
-E-001:INST.:  5      :DON.:21
**
```

(1) DON.:* signifie qu'aucune ligne données n'a encore été lue.

C) REPARTIR

RE

Action et bloc engendré au TI (cf Exemple)

Cette commande permet à l'utilisateur de reprendre l'éxécution du FSC là où elle a été arrêtée anormalement lors du dernier bloc associé à l'une des commandes suivantes :

EXECUTER, PAS A PAS ou TRACER.

Dans le cas où il n'y aurait pas eu de fin anaormale du dernier bloc associé à l'une des 3 commandes précédentes, un message est lancé au TI, indiquant que l'éxécution ne peut repartir, et un nouveau bloc est initialisé.

Dans le cas où il y a eu une telle fin anormale, les lignes texte du bloc engendré sont strictement semblables à celles qui peuvent apparaître dans des blocs EXECUTER, PAS A PAS ou TRACER.

Si une nouvelle fin anormale arrive, l'utilisateur est dans la même situation que précédemment (c'est à dire qu'il pourra à nouveau faire REPARTIR).

Si aucune fin anormale n'arrive avant que l'instruction <n2> du FSC ne soit éxécutée ou que le programme essaie de lire au delà de la ligne <n4> du FDC, le bloc se termine normalement. (Les numéros <n2> et <n4> sont ceux spécifiés dans la commande dont le bloc associé s'est terminé anormalement).

D) RANGER

RA

1°) Action

Cette commande permet à l'utilisateur d'affecter des valeurs à certaines variables et/ou à certains éléments de tableau de son programme courant autrement que par programme et sans autre modification de l'état du programme.

2°) Bloc engendré au TI.

L'utilisateur est sollicité par les IL numéro externe de ligne pour donner ses ordres d'affectation.

Ces numéros externes ne sont là que pour indiquer à l'utilisateur quand il peut taper sa ligne et n'ont aucune interférence avec les numéros externes des lignes du fichier source courant.

La numérotation recommence à 1 à chaque utilisation de la commande.

Formes des ordres d'affectation

Ces affectations peuvent être de 2 types :

- affectation de valeur à une variable sous la forme classique d'une égalité (cf Ex1).

La partie droite de l'égalité peut contenir des entiers, des réels, et des références à n'importe quelles variables ou fonctions du programme.

- affectation de valeurs aux éléments d'un tableau sous la forme d'un DO implicite (cf Ex2).

La partie droite peut contenir également des entiers, des réels, des références à des variables ou des fonctions du programme et des références à la variable contrôlée du DO implicite, qui peut ne pas être définie dans le programme.

Ces ordres d'affectation ne peuvent contenir de caractères de prolongation. De plus dès que le RC d'une affectation est tapé aucune modification ne pourra être apportées à cette ligne.

Chaque ordre est immédiatement éxécuté.

Une erreur décelée se traduit par la sortie d'un message et ensuite d'un nouveau numéro externe de ligne.

L'utilisateur peut arrêter la liste des affectations par un RC isolé.

Un autre bloc est alors initialisé par **.

3°) Remarque pratique

L'utilisateur ne peut référencer en partie droite de ses affectations que des variables et/ou des tableaux auxquels ont été alloués des zones de mémoire. C'est donc qu'avant de lancer un RANGER, il faut qu'il ait lancé précédemment un ou plusieurs EXECUTER, PAS A PAS ou TRACER avec comme paramètres des séquences d'instructions <SI> contenant au moins une fois, chacune des variables et/ou chacun des tableaux référencés dans le bloc RANGER.

4°) Exemple

```
        ** RA
Ex1      1   :A=3.14+X*Y+TAB(J)
Ex2      2   :  T(K) = K/2 , K=1,4
         3   :  C=FONCTION(A)
         4   :   RC
        **
```

Commentaire

A,J,Y,X,C,T sont des variables du programme courant. K est la variable contrôlée du DO implicite.

FONCTION est une fonction du programme courant. Dans l'ordre d'affectation 3, A a la valeur qui lui a été affectée dans l'ordre 1.

E) IMPRIMER

IM

1°) Action

Cette commande permet à l'utilisateur d'imprimer au TI les valeurs de certaines variables et/ou de certains éléments de tableau de son programme courant autrement que par programme et sans aucune modification de l'état du programme.

2°) Bloc engendré au TI

L'utilisateur est sollicité par les IL numéros externes de ligne pour donner ses ordres d'impression immédiate. Comme pour la commande RANGER ces numéros externes n'ont aucune liaison avec ceux du fichier courant et la numérotation recommence à 1 à chaque utilisation de la commande.

Forme des ordres d'impression

Chacun des ordres nécessite deux lignes dont chacune a un IL particulier :

- la première : nnn :LISTE:
- la deuxième : FORMAT:

La première ligne permet de spécifier une liste d'E/S contenant tous les éléments à imprimer.

La deuxième permet de spécifier le format d'impression. La liste et le format doivent répondre strictement aux spécifications du Fortran de base.

Si l'utilisateur répond par un RC isolé à l'IL FORMAT:, un format par défaut sera pris pour la liste donnée (pour les réels E15.8, pour les entiers I11). Tous les éléments symboliques référencés dans la liste doivent être définis dans le programme.

Il n'est pas possible de taper de lignes de prolongation et aucune modification d'une ligne n'est possible après la frappe du RC correspondant.

Dès la frappe du RC de la ligne spécifiant le format, l'éxécution de l'impression commence et les lignes correspondantes sont envoyées au TI.

Lorsque l'impression est terminée (qu'il y ait eu erreur ou pas), l'utilisateur est sollicité pour spécifier un nouvel ordre d'impression.

Si l'utilisateur répond par un RC isolé à l'IL nnn :LISTE:, un autre bloc est initialisé.

3°) Remarque pratique

L'utilisateur ne peut référencer dans les listes d'E/S que des variables ou des tableaux auxquels ont été affectés des valeurs. C'est donc qu'une commande d'éxécution aura été lancée précédemment.

4°) Exemple:

```
** IMPRIM
  1    :LISTE:A,B,K
       FORMAT:2E5,2,12
                                    ligne d'impression
  2    :LISTE:(A(J),J=1,9)
       FORMAT: RC
                                    ligne d'impression
                                    avec format par défaut
  3    :LISTE: RC
**
```

E) COMMANDES CONCERNANT LA SORTIE DES RESULTATS au TI

PERFORER RESULTATS	PR
IMPRIMER RESULTATS	IR
PERFORER IMPRIMER RESULTATS	PIR

1°) Action et bloc engendré au TI

Ces commandes permettent à l'utilisateur d'envoyer sur la machine à écrire et/ou sur le perforateur de ruban du TI les résultats d'instruction WRITE du programme courant, dont l'unité périphérique associé est l'"imprimante".

Si l'utilisateur n'a tapé aucune commande concernant la sortie des résultats, les résultats éventuels de WRITE sur "imprimante" sont uniquement imprimés au T.

Pour les perforer avec ou sans liste imprimée simultanée, il est nécessaire de taper respectivement PIR ou PR.

Pour revenir à une impression simple il faudra taper IR. Si l'utilisateur tape une des 3 commandes qui correspond précisément à la manière dont sorte ses résultats, il n'y a aucun changement. (Par exemple si l'utilisateur tape IR alors que ses résultats sont simplement imprimés au TI).

Dans les blocs associés à ces 3 commandes il n'y a que la ligne commande.

Un nouveau bloc est initialisé immédiatement.

2°) Remarque

La frappe de cette commande peut intervenir à n'importe quel moment (même après un BREAK en cours de sortie de résultats).

G) TRACER

Les paramètres ont la même signification que dans la commande EXECUTER

1°) Action

Cette commande permet d'obtenir au TI l'impression dynamique des valeurs de certaines variables ou de certains éléments de tableau parallèlement à une éxécution normale. (C'est à dire une éxécution semblable à celle obtenue avec la commande EXECUTER).

2°) Bloc engendré au TI

L'utilisateur est d'abord sollicité par les IL numéro externe de ligne pour donner ses ordres d'impression ultérieure. Comme pour les commandes RANGER et IMPRIMER ces numéros externes n'ont aucune liaison avec ceux du fichier courant, et la numérotation recommende à 1 à chaque utilisation.

Dans le cas de TRACER, ces numéros serviront à référencer les ordres spécifiés par l'utilisateur

Formes des ordres d'impression

Chacun de ces ordres nécessite deux lignes dont chacune est caractérisée par un IL :

- la première par nnn :LISTE:
- la deuxième par POINTS:

La première ligne permet de spécifier une liste d'E/S contenant tous les éléments à imprimer. Cette liste doit être conforme aux spécifications fortran de base et ne peut contenir de signe de prolongation.

La deuxième ligne permet de spécifier les points du fichier source courant en lesquels l'ordre d'impression sera exécuté.

Cette spécification se fait sous la forme d'une liste de numéros externes d'instruction séparés par des virgules.

L'utilisateur dispose de plus du caractère spécial '&' qu'il peut taper n'importe où dans la liste à la place d'un numéro externe d'instruction. Ce caractère a pour signification : "toutes les ruptures de séquence" ; ceci revient à dire que l'ordre d'impression associé sera éxécuté chaque fois qu'il y aura une rupture de séquence.

Dans le cas de TRACER, l'utilisateur ne peut pas définir de format.

Enfin, tous les noms symboliques se trouvant dans la liste doivent être définis dans le programme courant.

L'utilisateur spécifie ainsi une suite d'ordres d'impression.

La liste s'arrête lorsque il répond par un RC isolé à un IL
nnn :LISTE:

A partir de ce moment là l'éxécution traditionnelle du programme courant est lancée par la PC et le bloc engendré peut comporter tout ce qui peut se trouver dans un bloc engendré par la commande EXECUTER (résultats, erreurs, pseudo-blocs DEFINIR FICHIER, pseudo-blocs DONNEES). Cependant, chaque fois qu'une instruction dont le numéro externe se trouve dans une des listes de POINTS vient d'être éxécutée, des lignes texte correspondant à l'ordre d'impression, viendront s'ajouter au bloc normal.

Forme des lignes texte constituant les résultats de la trace

Les lignes comportent un IL indiquant

- qu'il s'agit d'un résultat de trace (T).
- et à quel ordre d'impression cette ligne est associée (nn).

L'IL est de la forme : -E-Tnn:

A la suite de cet IL et dans la même ligne, est imprimé le numéro externe de la dernière instruction éxécutée. Cependant la place de ce numéro dans la ligne apporte une information supplémentaire :

Si dans le programme courant, le contrôle passe à un sous-programme ou à une fonction qui contient des instructions "point de trace", les numéros externes correspondant apparaissent au TI décalés dans la ligne de deux caractères vers la droite. Quand le contrôle revient au programme appelant, les numéros externes apparaissant dans des traces éventuelles sont décalés de deux caractères vers la gauche.

Il y a cinq niveaux d'imbrication possibles.

Enfin au-delà de cette zone servant à imprimer le numéro externe, et éventuellement sur plusieurs lignes, sont imprimés les résultats de la trace suivant un format fixe.

La terminaison du bloc est identique à celle de EXECUTER.

3°) Exemple

Supposons que le programme courant de l'utilisateur ait la structure suivante :

(les numéros sont les numéros externes des lignes).

```
1
2
3   GOTO            Programme
4                   principa
5
6   CALL
7
8
9   SUBROUTINE
10                  sous-
11                  programme
12
13  RETURN
```

Supposons de plus qu'au cours de l'éxécution les éléments du tableau IT défini dans le programme principal garde toujours la même valeur :1

Supposons de plus que les variables M et N gardent toujours la même valeur :2

```
** TR
  1   :LISTE:M,N
      POINTS:2,13,8,10
  2   :LISTE:(IT(J),J=2,5)
      POINTS:&
  3   :LISTE: RC
-E-TO1:  2.          2 2
-E-TO2:  3.          1 1 1 1
-E-TO2:  6.          1 1 1 1
-E-TO1:   10.        2 2
-E-TO1:   13.        2 2
-E-TO2:   13.        1 1 1 1
-E-TO1:  8.          2 2
**
```

(1)

(1) Toutes les lignes suivantes sont envoyées par la PC.

Commentaires

1 - Entre ces lignes de résultat de trace, auraient pu apparaître des lignes textes (résultats, erreurs, etc...) dues à l'éxécution traditionnelle du programme.

2 - Dans les lignes de résultats de trace, les numéros externes de ligne apparaissent avec un '.' même si ce sont des entiers (comme dans l'exemple) afin de faciliter le repérage dans la ligne.

H) DESACTIVER TRACE et ACTIVER TRACE

DT

AT

1°) Action

Ces 2 commandes ont des effets complémentaires.

La première sert à désactiver momentanément certains ordres de trace spécifiés dans la dernière commande TRACER lancée, à condition que le bloc associé se soit terminé de façon anormale (erreur ou BREAK). La deuxième au contraire sert à réactiver certains des ordres de trace désactivés.

2°) Bloc engendré au TI

Après avoir lancé une de ces deux commandes, l'utilisateur est sollicité pour fournir une liste de numéros externes d'ordre de trace par l'IL LISTE:. Les numéros externes sont séparés par des virgules. L'utilisateur a également la possibilité de taper le caractère spécial '&' à la place d'un numéro externe ce qui a pour effet de désactiver ou d'activer tous les ordres de trace spécifiés lors de la commande TRACER.

Lorsque l'utilisateur a tapé le RC clôturant la liste, le PC initialise un nouveau bloc.

3°) Remarque

Ces deux commandes ne relancent pas l'éxécution. Si l'utilisateur veut relancer une trace anormalement terminée, il doit taper REPARTIR.

Remarque :

La présentation des commandes qui vient d'être faite correspond à un travail en collaboration étroite avec Monsieur Delpuech. Elle figure dans ce document pour situer l'ampleur du produit et justifier l'adoption d'une stratégie de réalisation rigoureuse.

Les travaux de Monsieur Delpuech préciseront les démarches pour le choix du jeu de commande.

2.2.2. La phase des spécifications de réalisation :

On entre dans le cadre de la stratégie choisie. A ce stade on opère un partage du travail. Pour chaque étape il faut donc indiquer la nature du travail et la personne qui en sera responsable. L'ordre de présentation est chronologique.

2.2.2.1. Spécification du niveau I(RU) :

Il est intéressant de confier les routines utilitaires indépendantes du mode à la personne chargée de rendre la machine initiale apte à gérer une partition conversationnelle (1). Les routines utilitaires du mode fichier seront confiées à la personne chargée de réaliser les commandes du mode fichier.

Pour établir ces routines et leurs interfaces avec les SPTC, on adopte la même démarche que lorsqu'on définit la gamme des instructions d'une machine. En effet, il s'agit simplement de rendre disponibles à partir du fortran, certaine des instructions assembleur en tenant compte du mode d'adressage de la mémoire virtuelle.

2.2.2.2. Spécification du niveau 3(MTC) :

Cette partie sera entièrement confiée à la même personne car l'enchaînement doit être homogène et il s'agit de construire un outil de pilotage de l'ensemble des fonctions. (2). Cette phase permet de recenser les SPTC qui seront nécessaires et les informations à transmettre.

2.2.2.3. Spécification du niveau 2 (SPTC) :

La phase précédente a déblayé le travail. La répartition se fait de la façon suivante : - l'analyse génération et l'interprétation sont confiées à une équipe de deux personne (3). La personne chargée du niveau des MTC donnera des contraintes de modularité et définira la zone d'influence du compilateur et de

---

(1) Cf. Travaux de Monsieur Delpuech

(2) Cf. Spécifications de réalisation C.I.I. Cette partie a constitué le plus gros de mon travail pendant le contrat.

(3) 2 élèves de 3ème année

l'interprèteur. Elle indiquera également la localisation des différents pointeurs à utiliser.

- Tous les autres SPTC sont spécifiés par la personne qui s'est occupée des MTC.(2).

2.2.3. La programmation et les tests

2.2.3.1. La programmation

Les différentes parties sont programmées par les personnes qui les ont spécifiées. Ce n'est pas une obligation mais seulement un problème de personnel.

La stratégie de réalisation se modifie car tous les interfaces sont convenablement étudiés à ce stade.

Une méthode de "bas en haut" semble très interessante car elle fait appel à une habitude de programmation.

En effet lorsqu'on programme dans un langage on travaille à partir d'un jeu d'instructions.

Ici à un niveau donné on fait la même démarche puisque la gamme d'instruction est l'ensemble des fonctions du niveau inférieur. On assiste à une définition dynamique de différents "jeu d'instructions".

Cette méthode est très agréable car on ne mélange pas les préoccupation. Dès qu'on a dépassé le numéro 1 on travaille uniquement en langage évolué et avec un degré de détachement croissant du niveau physique.

En ce qui concerne les notations et les conventions on impose une normalisation très stricte.

En même temps que l'on écrit un module on met à jour un index de correspondance. (Par exemple tous les RU sont identifiés par RUnnn ou nnn et un nombre. L'index indique que RU95 est un module de consultation de répertoire).

---

(2) Cf. Spécification de réalisation C.I.I. Cette partie a constitué le plus gros de mon travail pendant le contrat.

2.2.3.2. La phase de tests :

Elle s'est déroulée d'une manière assez particulière pour une raison matérielle : il n'y avait pas d'iris 50 à Grenoble. Or les 3 premiers niveaux ne peuvent au moins dans leur version opérationnelle n'être testés que sur IRIS 50. Cependant il était possible de simuler une grande partie des 3 niveaux inférieurs sur le matériel IBM dont nous disposions à Grenoble. C'est ce qui a été fait(I).

Dès lors que l'on disposait des niveaux inférieurs les SPTC et les MTC pouvaient être testés sur n'importe quelle installation disposant d'un fortran. Dans ces conditions il était intéressant de choisir l'installation fournissant l'environnement de mise au point maximum. Cp/cms fournissait un environnement conversationnel très intéressant pour les raisons suivantes :

- une grande facilité pour simuler le dialogue
- une grande rapidité de mise au point de programmes FORTRAN grâce à l'éditeur.
- un temps machine disponible important.

Remarque :

Il est clair qu'un des avantages de la stratégie adoptée réside dans la possibilité de faire les remises en questions nécessaires au plus tôt. En outre il est très agréable de ne pas mélanger des préoccupations différentes au même instant : lorsqu'on est à un niveau on n'a ici que les préoccupations inhérentes à ce niveau. (Ceci est fondamental pour l'agrément et la répidité des tests qui constituant une contrainte très forte). (2).

---

(1) Le seul mode qui ne pouvait pas être réalisé ainsi est le mode fichier car il aurait fallu fournir un effort de simulation disproportionné avec le gain.

(2) Sur un contrat d'une durée totale de 15 mois le temps s'est à peu près réparti de la façon suivante : 7 mois de conception comprenant toutes les spécifications générales, 4 mois de programmation, 4 mois de tests comprenant les tests définitifs sur IRIS 50 et les tests de réentrance.

A) Le matériel utilisé

a) Le matériel physique

Les installations de la C.I.I. aux Clayes-sous-Bois et l'installation de l'I.M.A.G. à Grenoble pouvaient être utilisées.

Ceci fournissait :

- d'une part deux prototypes IRIS 50 fonctionnant sous SIRIS II dont un a été muni de deux télétypes. Le dialogue avec les télétypes est celui défini par la partition conversationnelle uniquement. Les langages disponibles étaient le FORTRAN et le langage ASSIRIS

- d'autre part un 360/40 et un 360/67 disposant de CP/CMS. Les langages disponibles étaient le FORTRAN, l'assembleur 360 et le langage de commande de CMS.

b) Le temps machine disponible

Sur l'installation C.I.I. on pouvait espérer une heure à deux heures par jour pour travailler à partir des télétypes et un à deux passages de programmes en travail série. (Les télétypes n'ont été disponibles qu'à partir du mois d'octobre 1970).

Sur l'installation de Grenoble on pouvait compter 4 à 5 heures par jour d'utilisation CP/CMS avec 2 machines virtuelles indépendantes. Ceci permettait de tester les programmes utilisant un environnement simulé et de procéder à leurs corrections en mode conversationnel.

c) Le matériel humain

L'équipe se composait de deux ingénieurs et de deux élèves de 3ème Année de l'I.P.G.

Les deux ingénieurs possedaient une expérience système et de programmation en FORTRAN, ASSIRIS et Assembleur 360, les deux élèves une expérience d'analyse syntaxique et de programmation Fortran.

Tout le monde savait utiliser CP/CMS.

B) Le déroulement des tests

Après s'être assuré que les RU écrits pour l'IBM 360 pourraient être traduits exactement pour IRIS 50, on a testé successivement les SPTC et les MTC. A ce stade, on assure la correction de la logique des programmes et de l'enchaînement. On ne peut pas tester la réentrance. Il faut attendre de disposer sur IRIS 50, des RU, du moniteur local et de deux télétypes. La phase des tests de réentrance constitue la partie terminale du travail.

En résumé, les tests des modes compilation et exécution ont été faits entièrement à Grenoble sous CP/CMS augmenté d'un environnement de mise au point spécial (1).

La partie fichier a été entièrement testée sur IRIS 50 ainsi que les RU définitifs. (2).

(1) cf. Monsieur DELPUECH

(2) cf. Seconde Partie.

DEUXIEME PARTIE :

UNE GESTION DE FICHIER CONVERSATIONNELLE

0. INTRODUCTION

Cette partie a une vocation multiple : illustrer les considérations théoriques de la première partie, décrire dans un cas concret les problèmes et les démarches d'une réalisation effective et enfin montrer comment à partir d'exigences issues du Fortran on a débouché sur une gestion de fichiers beaucoup plus complète.

Les différentes fonctions de la partition s'opèrent en zone de travail et en l'absence d'une sauvegarde dans des fichiers permanents, ce qui est élaboré ne pourrait pas dépasser le cadre d'une session. D'un autre côté, il est intéressant de pouvoir utiliser du source ayant une autre provenance que le téléimprimeur. Ces considérations ont induit l'idée d'une gestion de fichiers dont la principale fonction serait bien sûr de satisfaire les exigences du Fortran mais qui présenterait toutefois une certaine indépendance vis-à-vis de celui-ci.

Nous définissons ainsi un mode nouveau qui constitue une des originalités du système qui a été réalisé. Nous en présenterons successivement les spécifications de définition, de réalisation, puis après quelques remarques nous donnerons des indications sur la programmation et les tests.

I - SPECIFICATION DE DEFINITION DU MODE FICHIER

1.1 Les objectifs généraux

1.1.1 En mode compilation

Permettre la compilation d'un fichier source résidant sur disque constitué au cours de la session courante, d'une autre session, ou même en dehors de l'enceinte conversationnelle. Il semble intéressant d'identifier ce fichier par un nom. En outre il faut pouvoir définir ce fichier dynamiquement et le plus brièvement possible.

1.1.2 En mode exécution

Permettre de faire correspondre à l'identificateur présent dans les ordres READ/WRITE des programmes de l'utilisateur un fichier en l'identifiant dynamiquement en économisant au maximum le dialogue.

1.1.3 Mode fichier

C'est un mode qui est moins étroitement lié au Fortran. Ses objectifs sont les suivants :

- permettre les identifications citées ci-dessus,
- permettre de créer de nouveaux fichiers dynamiquement destinés à sauvegarder tout ou partie des fichiers source ou données courants élaborés en zone de travail,
- permettre une gestion de ces fichiers,
- répertorier tous les fichiers dont il a été question vis-à-vis de la partition.

1.1.4 Indépendamment des commandes

Par un souci de grouper des préoccupations semblables la gestion de fichier au sens général englobe tout ce qui concerne les fichiers de la partition elle-même (mémoires virtuelles, répertoires, fichier message, etc ...). Cette partie fera l'objet d'un appendice.

1.2 Définition des éléments du mode fichier

Dans un premier temps seront définis les types de fichiers qui seront utilisés et ensuite les commandes seront présentées.

1.2.1 Les différentes classes de fichiers

Le concept dominant a été celui de bibliothèque. Ces bibliothèques possèdent des fichiers membres dans lesquels seront sauvegardés les éléments de source destinés à une utilisation ultérieure.

La première idée a été d'attacher des bibliothèques à la partition elle-même et de les partager entre les différents utilisateurs. Les fichiers seraient créés et utilisés au cours des différentes sessions exclusivement. Ceci ne couvre pas tous les objectifs car on désire que les utilisateurs puissent utiliser du source contenu dans des fichiers créés en dehors de la partition. L'idée est de rendre utilisable par la partition des bibliothèques privées de l'utilisateur (créés en travail série par exemple). Il est clair que ceci pose le problème de la compatibilité des formats et de l'identification de ces nouvelles bibliothèques.

Quant aux fichiers utilisés par les ordres d'entrées/sorties Fortran ils possèderont la structure qui leur est propre et le problème sera de les rendre accessibles à la partition.

Les fichiers propres à la partition n'entrent pas dans le propos de la gestion conversationnelle, ils seront mentionnés en appendice soit pour l'originalité de leurs contenus soit pour permettre au lecteur de bien comprendre le fonctionnement de la partition. Ces derniers fichiers ne posent en effet aucun problème théorique.

En étudiant le problème plus à fond et en tenant compte de l'orientation disque de la partition l'organisation suivante a été définie :

L'utilisateur peut manipuler 3 classes de fichiers.

1.2.1.1 Classe 1

Cette classe comprend des fichiers séquentiels contenant des lignes source ou des lignes données et qui peuvent être gérés par l'utilisateur au moyen de certaines commandes.

De plus ces fichiers ne dépassent pas l'enceinte définie par :

- la zone de travail de l'utilisateur
- et le téléimprimeur (lecteur-perforateur de ruban LPR ou machine à écrire MAE).

Les fichiers de la ZT (FSC et FDC) sont référencés implicitement suivant le type de la commande.

Les fichiers perforés ou imprimés au TI sont référencés :

- par le caractère spécial % pour le LPR
- et par la valeur par défaut d'un paramètre "nom de fichier" pour la MAE.

La distinction, entrée ou sortie, est contenue implicitement dans le nom de la commande.

1.2.1.2 Classe 2

Cette classe comprend des fichiers séquentiels continus contenant des lignes source ou des lignes données qui peuvent être gérés par l'utilisateur au moyen de certaines commandes.

De plus ces fichiers, ne peuvent se trouver que sur disque, doivent appartenir à une bibliothèque, et doivent être référencés par nom.

Ces fichiers peuvent avoir diverses origines au moment de leur création :

- un ensemble de lignes source ou données du fichier courant,
- un ensemble de lignes provenant du TI (MAE ou LPR),
- un fichier de bibliothèque.

Enfin, ces fichiers peuvent avoir diverses destinations au moment de leur utilisation :

- une partie du fichier courant,
- une impression ou perforation au TI,
- l'élément de bibliothèque.

1.2.1.3 Classe 3

Cette classe comprend les fichiers séquentiels continus, contenant des données et/ou des résultats et qui sont gérés par l'utilisateur au moyen des instructions d'E/S Fortran se trouvant dans le programme courant.

Ces fichiers peuvent être attachés à différents supports :

- disque,
- ruban perforé,
- machine à écrire (MAE) du TI.

Les fichiers de la classe 3 doivent être référencés par leur index.

Les fichiers se trouvant sur disque doivent être référencés par un index standard défini dans les spécifications Fortran CII.

Pour les fichiers lus ou écrits au TI (LPR ou MAE), il faut distinguer deux cas :

Cas des fichiers en sortie au TI.

L'utilisateur doit, dans l'instruction WRITE correspondante, référencer un tel fichier par l'index standard de l'"imprimante". Ensuite par le jeu des commandes

IMPRIMER RESULTATS
PERFORER RESULTATS
et PERFORER IMPRIMER RESULTATS

Il peut constituer son fichier sur ruban perforé et/ou à la MAE du TI (1)

Cas des fichiers utilisés en entrée au TI.

L'utilisateur doit, dans l'instruction READ correspondante, référencer

(1) cf. présentation des commandes.

un tel fichier par l'idex standard du "lecteur de cartes".

Cependant lors de l'exécution d'une telle instruction, les enregistrements à lire ne le seront ni sur le lecteur de cartes, ni au TI, mais dans le fichier données courant de la ZT de l'utilisateur.

L'utilisateur a ainsi la possibilité de constituer son FDC avant l'exécution (cf. la commande DONNEES) ou durant l'exécution (cf. les pseudo-blocs DONNEES).

Dans les deux cas ce fichier peut être constitué à partir de fichiers de la classe 2 ou de fichiers provenant du TI (LPR ou MAE).

Remargue :

L'usage des idex standard de l'imprimante et du lecteur de cartes permettent de ne pas modifier les instructions d'E/S lorsque l'utilisateur exécute le même programme dans un environnement "batch".

L'utilisateur doit cependant tenir compte du fait qu'au TI, les enregistrements ont une longueur de 63 caractères en entrée et de 70 caractères en sortie.

1.2.1.4 Sécurité de l'information contenue dans un fichier supporté par un disque

Pour accroître la sécurité de l'information contenue par un fichier disque, l'utilisateur peut dans tous les cas définir un mot de passe lors de la création de ce fichier.

Cependant en ce qui concerne la destination ou l'utilisation du fichier quelques différences apparaissent suivant les différentes sortes de fichiers.

<table>
<tr><th>Fichier \ Action</th><th>Création</th><th>Destruction</th><th>Utilisation</th></tr>
<tr><td colspan="2">Classe 1</td><td colspan="2">L'utilisateur qui a créé le fichier peut y lire ou y écrire des information en donnant le mot de passe s'il en existe un</td></tr>
<tr><td rowspan="3">Classe 2</td><td>BPB</td><td rowspan="3">N'importe quel utilisateur peut créer un fichier, avec ou sans mot de passe</td><td rowspan="3">L'utilisateur qui a créé le fichier peut le détruire en donnant le mot de passe s'il en existe un</td><td>L'utilisateur qui a créé le fichier peut l'utiliser en donnant le mot de passe s'il en existe un</td></tr>
<tr><td>BPC</td><td rowspan="2">Tout le monde, outtous ceux qui connaissent le mot de passe s'il en existe un, peuvent utiliser le fichier</td></tr>
<tr><td>BCC</td></tr>
</table>

1.2.2 Présentation des commandes fichiers

L'ensemble des commandes de gestion de fichiers se décompose en deux groupes.

- Commandes de définition des fichiers de la classe 2 et de ceux de la classe 3 qui sont sur disque

DEFINIR FICHIER qui permet à l'utilisateur de définir un fichier géré au niveau des instructions d'E/S Fortran (classe 3).

DEFINIR FICHIER BIBLIOTHEQUE qui permet de définir un fichier géré par l'utilisateur au niveau du langage de commandes (classe 2).

- Commandes de manipulation des fichiers (de classe 1 et 2)

SAUVEGARDER qui permet de ranger dans l'une des deux bibliothèques conversationnelles, dans une bibliothèque "batch" ou sur ruban perforé, tout ou partie du FSC.

AJOUTER qui permet de concaténer à un fichier, élément de bibliothèque ou perforé sur ruban, tout ou partie du FSC.

SAUVEGARDER DONNEES Commandes analogues aux précédentes mais qui utilisent le FDC.

DUPLIQUER qui permet de dupliquer un fichier, élément de bibliothèque ou envoyé au TI.

SUPPRIMER qui permet de supprimer un fichier d'une bibliothèque.

CATALOGUE qui permet d'obtenir la liste des noms des fichiers éléments d'une bibliothèque.

1.2.3 Description des commandes

1.2.3.1 DEFINIR FICHIER

DF<id>

Paramètre	Valeur du paramètre spécifié	
<id>	idex d'un fichier disque de classe 3	Ce paramètre est obligatoire

1°) Action

Cette commande permet à l'utilisateur d'identifier un fichier disque de classe 3 c'est-à-dire un fichier qui est géré au moyen des instructions d'E/S Fortran (READ, WRITE) du programme courant.

Cette identité est nécessaire pour que la PC puisse, soit créer de manière standard (c'est-à-dire au sens SGF) soit retrouver ce fichier déjà créé de manière standard.

2°) Bloc engendré au TI

A la suite de la ligne commande, 5 lignes texte sont imprimées au TI pour demander les renseignements concernant l'identité du fichier dont l'idex est <id>.

Chacune de ces lignes a un IL.

Les 5 IL sont, par ordre d'apparition, avec en regard les caractéristiques de la réponse que doit donner l'utilisateur :

NOM FICHIER :	1 à 17	caractères alphanumériques
IDENT VOLUME :	6	" "
NOM PROPRIETAIRE :	1 à 14	" "
MOT PASSE VOLUME :	1 à 10	" "
MOT PASSE FICHIER :	1 à 10	" "

Identification

- En vue d'une création de fichier : seul le premier argument est obligatoire. C'est-à-dire que l'utilisateur peut répondre par des RC isolés aux quatre autres IL.

- En vue de l'utilisation d'un fichier : tous les arguments spécifiés lors de la création de ce fichier, sont obligatoires.

Si l'utilisateur en spécifie moins, le fichier est incomplètement identifié; s'il en spécifie plus, alors que les paramètres obligatoires sont corrects, le fichier est parfaitement identifié.

En règle générale, un argument (obligatoire ou facultatif), non spécifié n'entraîne aucune réaction de la PC dans le cadre du bloc DEFINIR FICHIER. Par contre, un argument (obligatoire ou facultatif) qui est spécifié mais de manière incorrecte entraîne de la part de la PC l'impression d'un message d'erreur et la répétition de l'IL correspondant. Lorsque l'utilisateur a répondu aux 5 IL, un nouveau bloc est initialisé.

3°) Cas où l'utilisateur lance l'exécution du programme courant contenant des instructions d'E/S Fortran sur disque

Tous les fichiers disque de classe 3 doivent être définis par DEFINIR FICHIER.

Deux cas sont possibles :

1) L'utilisateur a défini tous ses fichiers avant de lancer l'exécution; auquel cas l'exécution des instructions d'E/S se déroule normalement.

2) L'utilisateur n'a pas complètement identifié un fichier disque; auquel cas, lors de l'exécution d'une instruction d'E/S référençant ce fichier, la PC va engendrer au TI un pseudo-bloc DEFINIR FICHIER.

La procédure alors activée est tout à fait semblable à celle qui concerne le manque données dans le FDC.

Il est à remarquer cependant que seulement les arguments non spécifiés et obligatoires de la commande DF sont demandés à l'utilisateur. Lorsque tous les arguments devant être spécifiés le sont, l'exécution repart et l'utilisateur en est averti par un message.

4°) Remarque

Les 5 arguments demandés sont seulement les arguments principaux d'identification. Une identification plus précise pour une utilisation en environnement "batch" devra se faire par carte avec l'ordre SGF .LABEL.

1.2.3.2 DEFINIR FICHIER BIBLIOTHEQUE

DFB<NOM>

Paramètre	Valeur du paramètre spécifié	
<NOM>	Nom d'un fichier de classe 2	Ce paramètre est obligatoire

1°) Action

Cette commande permet à l'utilisateur d'identifier le fichier de classe 2 spécifié par la valeur de <NOM> soit en vue de sa création soit en vue de son utilisation en tant qu'élément d'une certaine bibliothèque.

2°) Bloc engendré au TI

Les lignes texte qui apparaissent après la ligne commande dépendent de la bibliothèque concernée.

Ces lignes texte ont pour objet de demander les renseignements concernant l'identité du fichier.

Lorsque tous les IL ont eu une réponse, un nouveau bloc est initialisé au TI.

Cas d'un fichier élément d'une bibliothèque conversationnelle.

L'utilisateur est sollicité par un seul IL :

MOT PASSE FICHIER :

La réponse peut comporter au maximum 10 caractères alphanumériques.

- En vue d'une création : cet argument est facultatif.
- En vue d'une utilisation : cet argument est obligatoire s'il a été spécifié lors de la création.

Cas où l'utilisateur n'a pas lancé de commande DFB avant la création ou l'utilisation effective du fichier :

- dans le cas de la création, l'utilisateur n'est pas sollicité pour définir un mot de passe.
- dans le cas de l'utilisation, il est uniquement sollicité pour donner le mot de passe s'il en a été défini un à la création. Il n'y a donc pas d'apparition de pseudo-bloc DFB dans les 2 cas précédents.

Cas d'un fichier élément d'une bibliothèque "batch"

L'utilisateur est sollicité par 5 IL qui sont, par ordre d'apparition, avec en regard les caractéristiques de la réponse à donner :

NOM BIBLIOTHEQUE :	1 à 17	caractères	alphanumériques
IDENT VOLUME :	6	"	"
NOM PROPRIETAIRE :	1 à 14	"	"
MOT PASSE VOLUME :	1 à 10	"	"
MOT PASSE FICHIER :	1 à 10	"	"

Toutes les caractéristiques d'emploi de cette commande sont comparables, dans ce cas, à celles décritent en 1.2.3.1 pour la commande DEFINIR FICHIER. La seule différence étant que pour DFB, les fichiers sont gérés au niveau des commandes tandis que pour DF ils le sont au niveau des E/S Fortran. L'apparition au TI de pseudo-blocs DFB sera donc possible si l'identification du fichier est incomplète.

3°) Remarque concernant les fichiers de classe 2

Toute demande de création spécifiant un nom de fichier de classe 2 déjà créé entraîne l'impression d'un message d'erreur et l'initialisation d'un nouveau bloc.

Les seules commandes admises, concernant un fichier déjà créé, sont les commandes d'utilisation.

1.2.3.3 SAUVEGARDER

S<NOM>, <SI>

Paramètre	Valeur du paramètre spécifié	Valeur du paramètre par défaut
<NOM>	Nom de fichier	Ce paramètre est obligatoire
<SI>	Séquence d'instructions du FSC	Tout le FSC

1°) Action

Cette commande permet soit de sauvegarder la séquence d'instructions du FSC spécifiée par <SI> dans un fichier membre d'une bibliothèque que l'on

crée avec le nom <NOM>, soit de la perforer au LPR du TI.

La bibliothèque concernée dépend de la valeur du premier caractère de <NOM>. (* , . ou caractère alphanumérique).

2°) Bloc engendré au TI

SAUVEGARDER est une commande de création de fichier à partir d'un fichier de classe 1.

Si l'utilisateur s'en sert pour créer un fichier de classe 2, deux cas peuvent se présenter :

- la création a lieu dans une bibliothèque conversationnelle et rien n'est demandé à l'utilisateur.
- la création a lieu dans une bibliothèque "batch" et il peut apparaître un pseudo-bloc DFB si le fichier est incomplètement identifié.

Quand la sauvegarde a eu lieu, un nouveau bloc est initialisé.

1.2.3.4 AJOUTER

A<NOM>, <SI>

Les paramètres ont la même signification que pour SAUVEGARDER.

Action et bloc engendré au TI

Cette commande permet de concaténer au fichier <NOM> déjà créé la séquence du FSC spécifiée par <SI>.

AJOUTER est une commande d'utilisation du fichier <NOM>.

Cette différence mise à part, AJOUTER a les mêmes caractéristiques techniques que SAUVEGARDER.

1.2.3.5 SAUVEGARDER DONNEES et AJOUTER DONNEES

SD<NOM>, <SL>

AD<NOM>, <SL>

Paramètre	Valeur du paramètre spécifié
<NOM>	Nom d'un fichier de classe 2

Ce paramètre est obligatoire

1°) Action

Cette commande supprime du répertoire correspondant le fichier <NOM>.

2°) Bloc engendré au TI

Si le fichier à détruire est incomplètement identifié l'utilisateur devra dans le cas d'une bibliothèque conversationnelle fournir le mot de passe ou répondre aux IL d'un pseudo-bloc DFB dans le cas d'une bibliothèque "batch".

1.2.3.8 CATALOGUE

CA<B>

Paramètre	Valeur du paramètre spécifié	Valeur du paramètre par défaut
<B>	* (répertoire de la BPC) . (répertoire de la BCC)	Répertoire des BPB

Action et bloc engendré au TI

Suivant le caractère tapé comme valeur de <B>, c'est le répertoire d'une certaine bibliothèque qui sera listé au TI. La liste terminée, un nouveau bloc est initialisé.

Ces commandes sont analogues à SAUVEGARDER et à AJOUTER à ceci près qu'elles élaborent des fichiers données en bibliothèque ou sur ruban à partir du FDC.

1.2.3.6 DUPLIQUER

DU<NOM1>, <NOM2>, L

Paramètre	Valeur du paramètre spécifié	Valeur du paramètre par défaut
<NOM1>	Nom de fichier	Fichier à la MAE du TI
<NOM2>	Nom de fichier	Fichier à la MAE du TI
L	"avec liste au TI"	"sans liste au TI"

1°) Action

Cette commande permet de dupliquer un fichier <NOM> membre d'une bibliothèque ou provenant du TI, en un fichier <NOM2> membre d'une bibliothèque ou envoyé au TI.

La seule condition restrictive c'est que : <NOM1> ≠ <NOM2> (qu'ils soient spécifiés ou par défaut).

2°) Bloc engendré au TI

Avec cette commande, l'utilisateur peut utiliser et/ou créer des fichiers de classe 2.

C'est donc qu'un mot de passe peut être réclamé à l'utilisateur pour le fichier <NOM1> (ceci dans le cas d'une bibliothèque conversationnelle), ou qu'un ou deux pseudo-blocs DFB pourront apparaître au TI en cas d'identification incomplète (et ceci pour une bibliothèque "batch").

Lorsque la duplication est terminée un nouveau bloc est initialisé.

1.2.3.7 SUPPRIMER

SU<NOM>

2 - SPECIFICATIONS DE REALISATION DU MODE FICHIER

2.1 Recensement des routines des 3 niveaux

Dans ce paragraphe, nous nous proposons de recenser conformément à la stratégie générale de réalisation les RU, les SPTC puis les MTC sans entrer dans le détail des problèmes techniques qui seront étudiés en détail ultérieurement.

2.1.1 Les routines utilitaires

Elles correspondent aux fonctions générales à opérer pour manipuler les fichiers que nous avons définis plus haut. On distingue :

- l'allocation physique d'un fichier ou d'une bibliothèque
- l'assignation d'un périphérique à un fichier ou à une bibliothèque
- l'identification par LABEL d'un fichier ou d'une bibliothèque
- l'ouverture et la fermeture de fichier
- les entrées/sorties sur les fichiers

  GET1

  PUT1
- le repérage des fichiers comprenant les recherches en répertoire et les mises à jour.

2.1.2 Les sous-programmes de traitement de commandes

Ils correspondent à des macro opérations logiques liées au type du système :

- DF Cette routine collecte au téléimprimeur les renseignements permettant la création et/ou la définition et/ou l'identification d'un fichier de classe 3. Elle contient évidemment les 3 premières routines utilitaires.
- DFB Elle effectue le même travail pour les fichiers classe 2.
- OPEN et CLOSE effectuent l'ouverture et la fermeture d'un fichier quelque soit son type.

- GET2 permet la lecture de la carte initiale et des cartes suites dans un fichier source ou données et les range dans les fichiers courants correspondants suivant le format adopté pour les fichiers courants.
- PUT2 effectue l'opération inverse c'est-à-dire écrit sur fichier en normalisant sous forme initiale et suite une entrée du fichier source ou donnée courant.
- REC2 permet le rangement dans le FSC ou FDC de lignes frappées au TI.
- SEND2 permet de lister à la MAE ou de perforer au LPR des entrées du fichier courant (FC).

2.1.3 Les modules de traitement de commande

Ils correspondent aux commandes du mode fichier et on peut vérifier qu'avec les deux niveaux précédents on peut par un enchaînement convenable atteindre les objectifs de ces commandes. Leurs noms sont les mêmes que ceux des abréviations retenues pour les commandes précédés de MTC. (Par exemple, pour DUPLIQUER, le MTC est MTCDU).

2.2 Les problèmes et leurs solutions

2.2.1 Généralités

Les problèmes proviennent des contraintes qui nous sont imposées :

- contraintes de place : il s'agit d'occuper le minimum de place en mémoire centrale donc d'utiliser des modules, les plus courts et segmentés possible. En particulier, il faut éviter au maximum d'utiliser les modules du système qui sont chargés en même temps qu'un programme (modules d'accès du sgf par exemple). Il est également nécessaire que les modules soient formés de segments pouvant être gérés par la partition elle-même.

- contraintes de temps : la brièveté du temps de réponse est un élément très important. Plus encore il ne faut pas que l'utilisation d'une fonction longue par un des uti-

lisateurs grève le temps de réponse des autres. Or comme la partition constitue un programme utilisateur vis-à-vis du moniteur de SIRIS II, toute fonction moniteur mise en oeuvre directement sans passer par l'intermédiaire du multitasking (SGT) interdit la simultaneité.

- contraintes de compatibilité avec le software existant : cette compatibilité se situe essentiellement au niveau du format des fichiers et des bibliothèques qui doivent être manipulées en dehors de la partition par des programmes de SIRIS II. (Ces programmes par contre, ne peuvent pas être mis en oeuvre depuis la partition).

- contraintes imposées par le multitasking du SGT : elles interviennent au plan de la réentrance et de l'utilisation simultanée par plusieurs utilisateurs d'un même fichier. Ces situations qui peuvent être résolues par la notion de sous-programme protégé sont toutefois à éviter car elles suppriment la simultanéité sur laquelle s'appuie le fonctionnement de la partition.

Il apparaît clairement que tous ces problèmes concernent uniquement le niveau 1 et une fois qu'ils seront résolus c'est-à-dire une fois que les RU seront convenablement écrites, la réalisation ne pose plus de problèmes théoriques.

Nous allons maintenant passer en revue les problèmes en indiquant quelles solutions on leur a apporté et pourquoi.

Pour comprendre les problèmes et les solutions qui leur ont été apportées, il est nécessaire d'exposer sommairement le fonctionnement du système de gestion de fichier (SGF) de SIRIS II. Chaque fichier utilisé doit faire l'objet d'une description par macro instruction SGF qui crée une table de description de fichier (TDF) contenant 2 types de renseignements :

- une zone indépendante du type de fichier qui contient les paramètres et zones de manoeuvre nécessaires à la gestion de l'organisation support.

On appelle cette zone la zone OPEN/CLOSE.

- une zone qui dépend du type et liée à l'organisation fichier.

Pour compléter ou modifier ces tables, on dispose d'ordres sgf qui évidemment sont traités après ces descriptions. Enfin, le programmeur dispose de macro-instructions permettant l'ouverture et la fermeture d'une part et l'appel des modules d'accès d'autre part.

Ce qui importe pour notre propos, c'est de voir le travail et les figeages qui s'opèrent au cours des différentes phases d'assemblage, d'édition de liens et de chargement.

ASSEMBLAGE

Traitement des macros : il y a substitution des séquences générées.

Traitement des descriptions : on distingue les descriptions de groupement et les description de fichier.

La première réserve deux mots dont le premier contient le type de table et l'identificateur d'exploitation : idex. Le deuxième contient l'adresse d'une zone de manoeuvre.

Tous les fichiers du groupement pointeront sur ce mot.

Il y a en outre génération d'un article qui est associé au module objet contenant l'idex de FSD un identificateur et l'adresse des deux mots FSD dans le module objet.

La seconde réserve une table, y codifie les paramètres et génère un article qui est associé au module objet de la description du fichier.

Traitement des ordres SGF : Ils génèrent des articles associés en hors texte au modulent objet et rangent les arguments des ordres dans ces articles après une décodification éventuelle.

EDITION DE LIENS

A partir des articles FDL l'Editeur de liens crée un fichier de communication sur disque ainsi qu'un répertoire de ce fichier.

Il apparaît donc qu'à chaque article FDL correspond une zone de

communication du FILECOM caractérisé par l'idex de description FDL qui doit être unique dans le programme.

Le traitement des ordres SGF : consiste à modifier ou compléter les tables de description.

Après cette phase, il y a incorporation des modules SGF suite à une analyse du contenu des tables SGF.

Pour les tables TZP une zone de manoeuvre destinée à la gestion de l'allocation dynamique de la zone partagée est réservée dans le segment.

Pour les tables TDF l'éditeur détermine un module d'accès auto-adaptable et des critères d'adaptation. Si le module a déjà été incorporé dans un segment antérieur de la branche, l'éditeur place d'adresse du module déjà incorporé en TDF.

Sinon, il y a sélection adaptation et incorporation du module au segment de la table fichier et met l'adresse dans celle-ci.

Enfin, il y a incorporation au premier segment du programme d'un module de liaison chargé de faire les liaisons avec le moniteur.

CHARGEMENT

Le moniteur introduit en mémoire centrale le répertoire et place son adresse MC en renseignements généraux. Ce répertoire est résient en permanence dans la zone affectée au programme.

EXPLOITATION

A chaque ouverture fichier le SGF consulte le répertoire par balayage sur idex pour déterminer les zones filecom.

Il associe le fichier aux organes externes.

Il modifie éventuellement la table fichier.

Il exécute l'ouverture fichier en utilisant les renseignements LABEL.

Il élabore le programme de directives.

Il apparaît clairement que les problèmes vont naitre du caractère interpretatif que l'on souhaite pour la gestion des fichiers. En effet, il suffit de remarquer que les fonctions fichiers vont s'opérer dans la phase d'exécution du programme que constitue la partition toute entière. Il faut

donc pouvoir fournir dynamiquement les informations aux SGF afin de lui permettre de travailler. De son côté, le SGF impose des contraintes sur les informations minimum requises avant le chargement.

On retrouve certaines des contraintes signalées au 2.2.1 :

- contraintes générales d'utilisation d'un software existant
- contraintes de place pour les tables et modules rendus obligatoirement résidents
- contraintes de temps correspondant à la demande de fonctions moniteurs qui interdisent la simultanéité.

Nous étudierons maintenant les problèmes chronologiquement depuis la définition d'un fichier jusqu'à son utilisation.

2.2.2 Résolution des problèmes

2.2.2.1 Description des fichiers

Tout fichier est manipulé du point de vue utilisation par l'intermédiaire de sa table de description (TDF). Il doit donc correspondre une TDF à chaque fichier. Par ailleurs, il faut créer dans le FILECOM autant d'entrées différentes que de fichiers à traiter simultanément. L'importance de la description réside dans cette création d'entrée dans le FILECOM et dans l'incorporation des modules SGF auto-adaptable.

Le problème naît de l'impossibilité de connaître à la génération de la partition tous les fichiers susceptibles d'être utilisés au cours d'une session. L'utilisation des cartes moniteur est donc proscrite également.

Par contre, on peut utiliser une même TDF pour plusieurs fichiers de même type à condition de ne pas utiliser ces fichiers simultanément. (On entend par simultanément deux fichiers ouverts en même temps). Mais pour changer le fichier associé à une TDF, les opérations suivantes sont nécessaires :

- fermeture du fichier en cours (300 ms perdues pour tout le monde).
- lecture par macro RFCOM (ou plutôt son expansion) de l'entré du FILECOM.
- modification de cette entrée par les arguments du nouveau fichier

à l'aide d'une macro LABEL et ASSIGN.

- réécriture de l'entrée par macro WFCOM.
- ouverture du nouveau fichier (300 ms de perdues pour tout le monde).

Ces opérations, si on les répète trop souvent,risquent naturellement de dégrader considérablement les performances en rompant la simultanéité.

En résumé, une multiplication des TDF conduit à un encombrement mémoire important tandis que le partage des TDF détériore les performances. Il s'agit donc de trouver un critère qui permette de choisir un compromis entre ces deux contraintes extrêmes. L'idée est de permettre une utilisation normale rapide et peu encombrante. Ensuite on devra s'attendre (bien que les possibilités ne soient pas limitées) à ce que le fonctionnement se complique à mesure que les fonctions demandées seront moins classiques.

La solution adoptée est la suivante :

En premier lieu, il faudra une TDF de chaque type précisant les macro d'accès et ce, dans le but d'incorporer les modules d'accès auto-adaptables. Ensuite pour rendre les utilisateurs indépendants quant à l'utilisation d'un certain type de fichier, il faudra pour un type donné de fichier autant de TDF que d'utilisateurs. Pour un utilisateur donné, il faudra autant de TDF que de fichiers susceptibles d'être utilisés simultanément. Ces exigences étant connues, il reste à indiquer comment on peut y souscrire.

On désire comme indiquer plus haut manipuler simultanément les fichiers classe 3 d'une part et deux fichiers classe 2 (commande dupliquer). En limitant à 5 le nombre maximum des fichiers classe 3, il faudra donc par utilisateur :

- 5 TDF pour les classes 3
- 3 TDF pour les classes 2 (1 pour les membres de BPB, 2 pour ceux de BPC et BCC).

Ces TDF doivent exister à la génération de la partition et contenir les arguments qui doivent obligatoirement être figés par l'éditeur de liens. Pour ne pas encombrer la mémoire inutilement si ces TDF ne sont pas utilisées, on constitue 2 segments par utilisateur qui seront chargés au moment de leur

utilisation et seront vidés sous certaines conditions tenant à l'historique de la session. D'un point de vue utilisation, ces segments seront considérés comme des pages de numéro 256 et 257 et appelés par CALL BASE.

Le premier segment contient les TDF pour une utilisation en fonctionnement minimal. (Un manoeuvre Fortran, une BCC, une BPC, deux fichiers Fortran privés).Ce segment est chargé à la première exécution avec fichier et conservé en mémoire ensuite.

Le second segment contient une TDF pour un autre manoeuvre Fortran un fichier privé Fortran et une bibliothèque privée batch (BPB).

2.2.2.2 L'allocation

On distingue les fichiers séquentiels continus et les fichiers séquentiels chaînés. Les premiers sont ceux cités dans les ordres READ/WRITE des programmes Fortran et on ne doit procéder à une allocation que dans le cas des fichiers où l'on écrit et que l'on veut ainsi créer.

Dans tous les cas, l'allocation se fera dynamiquement par une macro LABEL d'allocation qui permettra d'effectuer l'allocation au moment de l'open. Pour les seconds on distingue encore deux cas :

- ceux qui sont membres d'une zone partagée ayant déjà fait l'objet d'une allocation (une zone partagée est allouée en bloc et non par fichier). Il n'y a alors rien à faire. C'est le cas des fichiers manoeuvres Fortran et des fichiers membres de BPC et BBC.
- ceux pour lesquels la zone partagée n'a pas été allouée. Il faut alors commencer par allouer la zone partagée par une macro LABEL spéciale d'allocation. On effectuera ensuite une macro LABEL pour le fichier membre sans albcation.

N.B. : Il faut noter que le mécanisme pour effectuer une macro label est un peu plus compliqué : il faut déterminer la TDF adéquate, lire l'entrée du FILECOM correspondante, la modifier par la LABEL et enfin réécrire cette entrée du FILECOM.

2.2.2.3 L'assignation d'un périphérique

Elle se fait toujours entre une lecture et une écriture de FILECOM par une macro ASSIGN.

2.2.2.4 L'opération LABEL

Elle s'effectue dans les mêmes conditions que l'assignation d'un périphérique mais au moyen d'une macro LABEL. Elle peut servir pour une identification seule ou pour une création accompagnée d'une allocation.

2.2.2.5 L'ouverture et la fermeture

Elles se font par l'expansion des macro OPEN et CLOSE complétées préalablement du fait du fonctionnement interprétatif.

2.2.2.6 Les E/S proprement dites

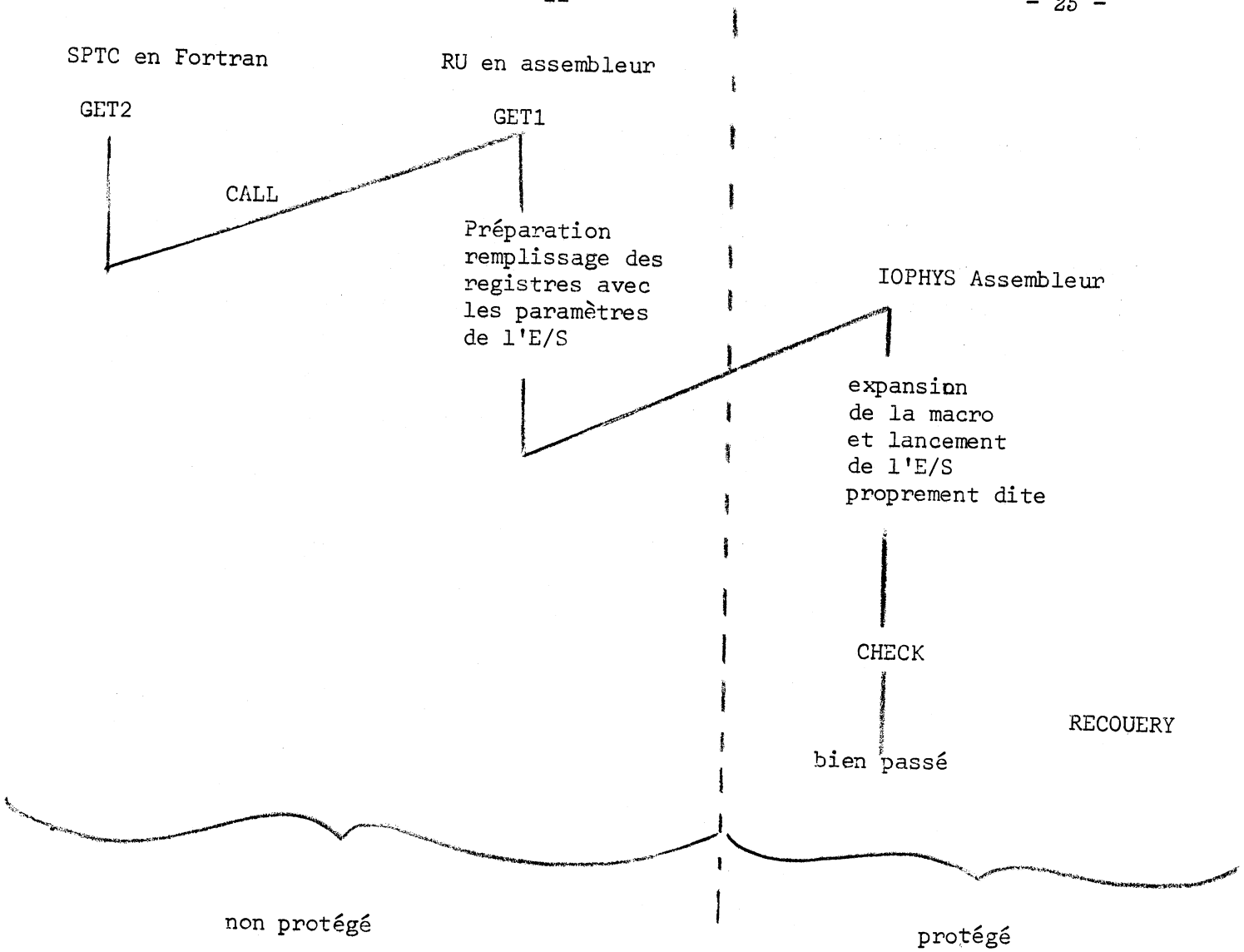
Le problème vient de l'utilisation en multitasking :

Les sous-programmes d'entrée/sortie sont écrits en langage d'assemblage et les zones de mémoires utilisées pour un utilisateur et locales au sous-programme risquent d'être détruites lors d'une perte de contrôle sur entrée-sortie. Il faut donc au maximum passer les paramètres par registres qui sont eux sauvegardés et restaurés automatiquement par le SGT.

Mais le problème le plus important reste l'utilisation simultanée d'un même fichier : par deux utilisateurs différents : une erreur SGF intervient lorsqu'on travaille au niveau bloc dûe au lancement de deux entrées/sorties non suivies d'un CHECK.

Le SGT fournit une solution en permettant de protéger des sous-programmes, c'est-à-dire de préciser qu'un utilisateur ne peut entrer dans un tel sous-programme que lorsque le précédent en est sorti. Il est évident que cette méthode si elle est pratique rompt la simultanéité : il convient donc de choisir avec soin les sous-programmes à protéger afin de rompre la simultanéité le plus tard possible et pour une durée la plus courte possible. Dans ce but, le mécanisme d'une entrée/sortie obéit au schéma suivant :

Prenons le cas d'une lecture de lignes source en bibliothèque



Remarque :

Le mécanisme et l'organisation d'une entrée/sortie disque qui vient d'être présenté est complètement identique à celui d'une entrée/sortie télétype. (cf. travaux de Monsieur Delpuech). En fait, pour ne pas compliquer la présentation on a volontairement omis un mécanisme annexe qui consiste à appeler des routines moniteur local pour permettre de prendre en compte le BREAK. Il s'agit des routines ATTENTE1 et ATTENTE2 nécessaires en raison de l'absence d'un ATTENTION hardware).

On remarque également qu'au niveau des SPTC la nature des mémoires auxiliaires est ignorée, puisque le même SPTC est utilisé pour le ruban ou l'un quelconque des fichiers disque. C'est la détermination de la TDF qui résoud ce problème à partir du nom fourni au TTY par l'utilisateur.

2.2.2.7 Les problèmes de compatibilité

Ils se posent chaque fois que la partition effectue des opérations présentant des analogies avec celles que l'on peut effectuer en travail série avec le software existant. Si on veut assurer la compatibilité, deux cas se présentent alors :

- ou bien on peut mettre en oeuvre les opérations séries à partir du conversationnel moyennant quelques modifications,

- ou bien on réalise une opération indépendante et on assure la compatibilité au niveau du résultat seul.

C'est le second cas qui se présentera le plus souvent et qui retiendra ici notre attention. La compatibilité intervient en mode fichier pour les points suivants :

- format des fichiers source et données sauvegardés en bibliothèque,
- structure des bibliothèques et répertoires,
- enfin, création des informations utiles pour une gestion ultérieure série.

Rappelons que SIRIS II possède un module de gestion des bibliothèques, le BIBLIOTHECAIRE, qui fonctionne de manière interprétative en utilisant les informations suivantes : pour chaque zone partagée NOMZONE constituant une bibliothèque, il crée un fichier membre particulier de nom NOMZONE/BIBREP qui sert de répertoire et qui est mis à jour lors de chaque opération. La partition qui ne peut pas inclure le module bibliothécaire doit donc gérer le fichier BIBREP après l'avoir créer au besoin. La méthode est alors la suivante : lorsqu'on utilise une bibliothèque, ou bien elle existe et il suffit de mettre à jour le répertoire BIBREP ou bien on la crée à partir de la partition et il faut en plus créer le répertoire BIBREP.

En outre, dans tous les cas, il faut prendre soin de produire des fichiers possèdant la même structure que ceux du Fortran série.

2.2.2.8 Le repérage des fichiers

Il est nécessaire de répertorier les fichiers vis-à-vis de la partition. Comme la durée de vie des fichiers manipulés dépasse la durée d'une session, il doit en être de même des répertoires. Nous utilisons donc des répertoires disques, c'est-à-dire des fichiers disque en accés direct. Recensons alors les informations nécessaires dans ces répertoires :

- tous les arguments LABEL recueillis
- tous les éléments nécessités par la partition (n° utilisateur, etc ...).

Or, puisque l'on veut fournir à l'utilisateur le maximum de facilité on lui a permis un repérage de ses fichiers par nom pour les fichiers bibliothèque et par idex pour les fichiers Fortran. Ce double repérage se retrouve dans la structure adoptée pour les répertoires. Elle s'allie à un souci de minimiser le nombre d'entrées/sorties lors des recherches. Il a donc été implanté 3 sortes de répertoires :

- un répertoire d'idex contenant principalement l'idex et un pointeur sur un répertoire général
- un répertoire de nom contenant le nom (vis-à-vis de la partition) et un pointeur sur le répertoire général
- un répertoire général pointé par les deux précédents qui contient toutes les autres informations utiles (zone label, etc ...).

L'intérêt de cette structure réside dans une préselection sur nom ou idex et dans la faible longueur des entrées des deux premiers répertoires qui permet de mettre beaucoup d'entrées dans un secteur disque (taille minima pour une entrée-sortie disque), limitant aussi le nombre des entrées-sorties nécessaires.

Un autre problème provient de la liberté laissée à l'utilisateur dans le choix du nom de ses fichiers. Ce nom ne peut pas être utilisé tel quel dans les opérations SGF (dans les macros LABEL par exemple) car on risque d'avoir deux noms identiques pour deux utilisateurs différents. Ceci oblige à adopter un double repérage partition-siris, c'est-à-dire qu'il faut manipuler un nom externe fourni pour l'utilisateur et un nom interne adopté par le SGF pour travailler sur le fichier. Pour caractériser complè-

tement un fichier on distingue deux cas :

a) Les fichiers créés en dehors de la partition

qui possèdent un nom partition et un nom propre qui constituent alors le nom interne.

b) Les fichiers créés au cours d'une session

le repérage externe est alors soit un nom soit un idex. Le nom interne est composé d'un préfixe et du numéro d'entrée dans le répertoire général utilisateur qui est évidemment unique. Les préfixes possibles sont :

BCC pour la bibliothèque commune
BPC " " " privée conversationnelle
BPB " " " " batch
FOR " " fichiers FORTRAN

Il apparaît que l'identification d'un fichier utilise les renseignements suivants :

le numéro d'entrée en RRU de l'utilisateur
le numéro d'entrée en RGF du fichier
le préfixe
le nom externe

La recherche d'un nom en répertoire se fait par balayage sur chacune de ces informations. (1)

(1) Le format des répertoires et les routines de consultation sont décrits en appendice.

Remarque :

La mise à jour des répertoires peut se faire de deux façons :

- de façon classique pendant une session à l'aide d'une routine utilitaire appropriée,
- exceptionnellement en travail série par utilisation du bibliothécaire et/ou de sous-programmes écrits à cet effet.

(Pour plus de détails, on se reportera à l'annexe décrivant les RU de consultation et de mise à jour des répertoires).

Nous avions vu dans la stratégie de réalisation l'importance du niveau I. La machine de niveau I vient d'être complétée par sa partie fichier. Elle rend maintenant disponible, aux niveaux supérieurs, une nouvelle machine qui remplace la machine primitive en offrant des possibilités supplémentaires et un plus grand agrément. Elle permet en outre de se dégager pour la suite de la réalisation du langage d'assemblage et de ne plus travailler qu'en Fortran. A partir du moment où l'on est assuré que les RU couvrent le domaine des possibles de leur niveau, on peut construire les niveaux supérieurs suivant la méthode qui semble la mieux adaptée.

Nous allons nous intéresser maintenant à ce problème.

2.3 Construction des niveaux 2 et 3

2.3.1 Les MTC

On passera en revue les différents MTC en indiquant seulement la liste des SPTC utilisés car les fonctions générales sont déjà présentées avec les commandes.

2.3.1.1 MTCDF

Il utilise essentiellement DF.

2.3.1.2 MTCDFB

Identique avec DFB.

2.3.1.3 MTCCA

Il permet de lister une partie des répertoires.

2.3.1.4 MTCS

Il utilise DFB, OPEN, PUT2, CLOSE et concerne le FSC.

2.3.1.5 MTCA

Mêmes sous-programmes que MTCS.

2.3.1.6 MTCSD et MTCAD

Font les mêmes choses que MTCS et MTCA sur le FDC.

2.3.1.7 MTCDU

Il utilise DFB, OPEN par abus de programmation GET1 et PUT1, CLOSE.

2.3.1.8 MTCSU

Il utilise la routine DELETE.

2.3.2 Les SPTC

Pour décrire les sous-programmes de traitement de commandes, on indiquera brièvement la fonction puis l'enchaînement des principales des routines utilitaires utilisées.

2.3.2.1 DF

Fonction : définir un fichier classe 3 pour permettre l'ouverture. Cette opération comprend la collecte de tous les arguments nécessaires à

partir du télé-imprimeur, l'opération label, la mise à jour des répertoires et des masques de définition.

Enchaînement des RU :

RCHIDX
ASSTDF
LABEL
RGIDX
RGRGF

2.3.2.2 DFB

Fonction : la même que pour DF mais relativement à un fichier classe 2.

Enchaînement des RU :

RCHNOM
ASSTDF
ALLZP
LABEL
RGNOM
RGRGF

2.3.2.3 OPEN CLOSE

Fonction : réaliser la fermeture et l'ouverture d'un fichier de classe 1 ou 2 indifféremment.

Enchaînement des RU :

OPENPHI

2.3.2.4 GET2

Fonction : lire dans un fichier classe 3 ou sur le ruban une instruction Fortran (ligne initiale et ligne suites) ou une carte donnée et la ranger dans le fichier courant correspondant. Gérer le pointeur dans le secteur courant.

Enchaînement des RU :

GET1

2.3.2.5 PUT2

Fonction : effectuer l'opération inverse de GET2.

Enchaînement des RU :

PUT1.

3 - LA PROGRAMMATION ET LES TESTS

La phase de programmation obéit à une stratégie différente de celle des spécifications de réalisation puisqu'à ce stade tous les éléments sont définis à un niveau assez fin pour permettre la programmation. On programme de bas en haut en définissant à chaque changement de niveau un nouveau langage dont les instructions sont les fonctions du niveau qui vient d'être programmé. Les seules entorses à la méthode proviennent soit de contraintes de programmation non entrevues auparavant (une étude s'impose alors), soit de problèmes de communications (paramètres mal spécifiés). Dans tous les cas, une remise en question ne concernera que le niveau voisin : en amont si on fait du haut en bas, en aval si on fait du bas en haut.

De toutes façons, on commence par assurer la correction définitive du niveau clé des routines utilitaires.

En ce qui concerne les tests, on distingue le Fortran et l'assembleur. On commence naturellement par l'assembleur qui correspond aux routines utilitaires et on teste les fonctions séparément sans tenir compte de la réentrance.

(Remarquons qu'avant de tester les routines utilitaires, on a déjà créé tout un environement fichier comprenant les répertoires, les fichiers propres à la partition ainsi que tout ce qui touche aux fichiers sans appartenir véritablement au traitement des commandes). La partie fichier a été testée complètement sur IRIS 50 car l'effort de simulation aurait été trop important.

La partie Fortran est ensuite écrite et testée indépendamment et sans tenir compte de la réentrance. (Pour celà, on utilise un seul télétype).

Quand tous les tests précédents sont reconnus satisfaisants, on teste l'enchaînement et la réentrance. On utilise alors deux télétypes et on effectue simultanément les mêmes opérations.

Les erreurs susceptibles d'apparaître peuvent être de deux types : des erreurs purement logiques que l'on détecte à l'aide du dump dynamique qui fait partie de l'environnement de DEBUG qui a été spécialement implanté (1) et des erreurs véritables de programmation qui donnent lieu à un dump

(1) cf. Travaux de Monsieur DELPUECH.

à l'imprimante lancé par les routines de reprise d'erreur de la partition. (Ce dump comprend toute la zone de mémoire occupée par la partition).

Au cas où on désire connaître l'état de la mémoire virtuelle sur disque, on utilise un DUMP statique de mémoire virtuelle qui liste toutes les tables en respectant les formats et la représentation utile (hexa ou ebcdic). (1).

Il s'agit ici des erreurs dont on pouvait être tenu responsable. On a rencontré des erreurs tenant à des parties du software de base incomplètement testées.

Notre travail utilisait en effet de nombreuses fonctions du software de base rarement utilisées en travail série.

Dans tous les cas, le DUMP général de la mémoire qui était demandé par la routine de reprise d'incident permettait de conclure (plus ou moins rapidement selon le responsable de l'erreur) et de s'adresser au responsable pour la correction de l'erreur. (Nous-mêmes, ou les responsables du software de base).

4 - REMARQUES SUR LA PORTEE DE LA GESTION DE FICHIER CONVERSATIONNELLE

Il s'agit d'étudier trois points : l'adéquation aux objectifs principaux, l'utilisation possible pour autre chose que la compilation et l'exécution de programmes, les extensions possibles et ce qu'elles nécessitent.

4.1 L'adéquation

On vérifie aisément que toutes les opérations nécessaires pour permettre la compilation et l'exécution de programmes Fortran sont disponibles dans le mode fichier. De plus, l'introduction des pseudo-blocs rend ces opérations relativement plus agréables qu'en utilisation série. Enfin, aucune limitation supplémentaire n'est imposée par l'utilisation de fichier en mode conversationnel par rapport à l'utilisation série.

(1) Les programmes statiques sont décrits en appendice.

4.2 L'utilisation en dehors de l'aspect Fortran

On peut utiliser la gestion de fichier conversationnelle pour créer et manipuler ensuite par les commandes d'édition de texte (MODIFIER) des fichiers contenant autre chose que du source Fortran à condition que certaines contraintes sur les formats soient respectées. Pour prendre un exemple, on peut manipuler des enregistrements de source assembleur. On crée un FDC avec les instructions assembleur, on modifie par la commande modifier données et on catalogue par la commande Sauvegarder Données. On peut ensuite Dupliquer dans un autre fichier ou au ruban.

Une autre utilisation très intéressante est la création et l'allocation d'un fichier et/ou d'une zone partagée (pour une utilisation série par exemple).

4.3 Les extensions possibles

On distingue celles qui se justifient à elles seules et celles qui se justifient dans le cas d'un changement de vocation de la partition. Les premières consistent à augmenter le caractère indépendant de la gestion de fichier et à réaliser une gestion de fichier complète conversationnelle. On peut penser à une modification du BIBLIOTHECAIRE de SIRIS II pour en faire un ensemble de RU. Il suffirait ensuite d'écrire un niveau de SPTG et peut-être de MTC pour une mise en oeuvre à partir du téléimprimeur. Cette extension se trouverait facilitée par la comptabilité partition bibliothècaire qui a déjà été respectée. On peut augmenter le nombre de types de bibliothèques manipulées (source texte, IMT).

Les extensions qui sont liées à la vocation du système dépendent essentiellement des fichiers requis pour la nouvelle utilisation. Par exemple, l'utilisation d'un assembleur conversationnel demande la manipulation de fichiers binaire translatable et dans le cas d'un éditeur de lien de format IMT. (Dans ce cas, il est clair que les fonctions bibliothècaire sont suffisantes). On peut également envisager de réaliser des composants nouveaux pour lesquels les extensions passeraient par la réalisation préalable d'un nouveau système de gestion de fichier à caractère conversationnel. (Par exemple, un système d'interrogation de fichier).

La statégie à adopter pour réaliser ces extensions est décrite dans la première partie.

CONCLUSION

On pourrait s'étonner de la disproportion entre la première et la seconde partie. Il n'y a pas lieu de le faire. Ce partage reflète en effet l'intensité du travail de création et de recherche depuis la commande du produit. La durée totale du contrat étant de quinze mois, la durée des différentes étapes a été la suivante :

- 7 mois de définition et de spécifications se divisant en 2 mois environ pour la définition des objectifs et 5 mois pour les spécifications. Cette période comprend la mise en place de la stratégie de réalisation et emploie 2 ingénieurs et 2 élèves de dernière année de l'I.P.G.

- 4 mois de programmation employant les personnes précédentes.

- 4 mois de tests comprenant les tests de réentrance.

Il est alors bien clair étant donnée la masse de programmation (plus de 200 K octets) que c'est la première phase qui a demandé le plus de travail de recherche.

Ce découpage montre également l'importance de la première phase et de l'adoption d'une stratégie de réalisation rigoureuse. Sans un système de spécification très précis pour guider constamment la programmation et limiter l'impact d'une remise en question, on aurait obtenu un temps de programmation considérablement plus long. La brièveté de la seconde phase est aussi le résultat de la prise en compte, dès le début, du matériel à notre disposition (personnel, compétences, matériel proprement dit, temps d'utilisation possible par jour). La méthode adoptée, par la modularité qu'elle introduisait, permettait la meilleure répartition du volume de travail entre ces différents éléments.

La période de programmation a été très automatique; elle a permis d'enrichir une expérience de programmation et d'acquérir une connaissance du software de base pour la réalisation du niveau intermédiaire.

Je désirerais terminer sur certaines remarques concernant la réalisation d'un composant tel que notre système sur une machine non nue.

Le bon avancement des travaux à l'intérieur de notre produit reposait sur deux éléments :

- le respect de la part des exécutants des spécifications qu'ils avaient écrites ou qu'on leur avait imposées.

- l'assurance que les éléments produits par les autres seraient parfaitement testés au moment où on les utiliseraient. (Par exemple, au moment des tests de réentrance, on devait être certain que toute erreur proviendrait non pas du moniteur local, mais du composant dont on testait la réentrance).

En s'astreignant à une discipline rigoureuse, on peut instaurer cette confiance entre les 'niveaux' qui sont indépendants de la machine initiale c'est-à-dire au-dessus du niveau intermédiaire. Si ce souci n'a pas présidé à la réalisation du software existant, en particulier s'il existe un décalage entre les spécifications et les produits ou si ces produits sont incomplètement testés, la réalisation du niveau intermédiaire devient très pénible. Or c'est de ce niveau intermédiaire que dépend l'addition d'un composant important au système initial.

Il semble donc que pour la production d'un software complet sur une gamme de machines, il faille commencer par fournir quelques composants très sûrs et extensibles plutôt qu'un grand choix de composants imparfaits mal testés ou difficilement extensibles. Les critères de spécification de modularité et de généralité doivent être appliqués à ces composants initiaux plus qu'à tous autres.

APPENDICE 1

OPERATEUR	OP	A
PAS A PAS	PAP	E
PERFORER RESULTATS	PR	E
PERFORER IMPRIMER RESULTATS	PIR	E
RANGER	RA	E
REPARTIR	R	E
SAUVEGARDER	S	F
SAUVEGARDER DONNEES	SD	F
SUPPRIMER	SU	F
SVP	SVP	A
TEMPS UNITE CENTRALE	TUC	A
TOTAL	TO	C
TRACER	T	E

APPENDICE 2

APPENDICE N° 2 : TABLEAUX RECAPITULATIFS DE LA STRUCTURE ET DE LA GESTION DES FICHIERS

Gestion des fichiers ne dépassant pas l'enceinte de la zone de travail et du téléimprimeur (fichiers de la classe 1).

Structure

<table>
<tr><td>Arrivée
Départ</td><td>ZT</td><td>TI
MAE</td><td>TI
LPR</td></tr>
<tr><td>ZT</td><td></td><td colspan="2">Utilisation du fichier courant</td></tr>
<tr><td>TI
MAE</td><td rowspan="2">Elaboration du fichier courant</td><td colspan="2" rowspan="2">Liaison MAE, LPR</td></tr>
<tr><td>TI
LPR</td></tr>
</table>

Détail des commandes

<table>
<tr><td>Arrivée
Départ</td><td>ZT</td><td>TI
MAE</td><td>TI
LPR</td></tr>
<tr><td>ZT</td><td></td><td>L LD</td><td>S SD</td></tr>
<tr><td>TI
MAE</td><td>C D</td><td></td><td>DU</td></tr>
<tr><td>TI
LPR</td><td>C D</td><td>DU</td><td></td></tr>
</table>

Commentaire

Par exemple, par l'intermédiaire de la commande COMPILER (C) ou de la commande DONNEES (D), l'utilisateur peut élaborer le fichier courant

- qui se trouve dans la ZT
- à partir du lecteur-perforateur de ruban du téléimprimeur (TI-LPR).

Gestion des fichiers de classe 2

Structure

Arrivée / Départ	ZT	TI MAE	TI LPR	BPC	BCC	BPB
ZT				Utilisation d'un fichier de classe 1 et création d'un fichier de classe 2		
TI MAE						
TI LPR						
BPC	Utilisation d'un fichier de classe 2 et création d'un fichier de classe 1			Utilisation d'un fichier de classe 2 et création d'un fichier de classe 2		
BCC						
BPB						

Détail des commandes

Arrivée / Départ	ZT	TI MAE	TI LPR	BPC	BCC	BPB
ZT				S SD	S SD	S SD
TI MAE				DU	DU	DU
TI LPR				DU	DU	DU
BPC	C D	DU	DU	DU	DU	DU
BCC	C D	DU	DU	DU	DU	DU
BPB	C D	DU	DU	DU	DU	DU

Commentaire

Par exemple, par l'intermédiaire de la commande DUPLIQUER (DU), l'utilisateur peut créer un fichier de classe 2

- dans une bibliothèque "batch" (BPB)
- à partir d'un fichier se trouvant au LPR du TI.

APPENDICE 3

APPENDICE N° 3 : LES FICHIERS PROPRES A LA PARTITION

Ce sont tous les fichiers qui permettent de faire fonctionner la partition et qui ne sont pas des fichiers utilisateurs au sens de la gestion de fichier.

On distinguera pour chaque fichier la création, l'initialisation et les mises à jour ou manipulations.

La mémoire virtuelle :

On dispose d'un seul fichier pour tous les utilisateurs. Il est créé par cartes moniteur avec le type accés direct et réside sur disque. Chaque page constitue un enregistrement physique de longueur I/2K. Les adresses relatives des différentes pages (n x 256 en tout pour n utilisateurs) sont donc des multiples de 2. D'un point de vue logique on distingue n zones contigües comprenant chacune les 256 pages d'un utilisateur.

A l'initialisation d'une session il faut initialiser certaines zones virtuelles : table des enchaînements, tables de compilation et d'interprétation. Une routine utilitaire correspond à cette fonction. Elle établie par programme le chaînage initial et initialise directement sur disque.

La lecture et l'écriture d'une page sur disque se font à l'aide de 2 routines utilitaires APAGE et RGPAGE. Dans les deux cas les paramètres sont le numéro de la page et l'adresse en mémoire centrale de la page. Les sous-programmes déterminent à partir du numéro d'utilisateur courant et des paramètres précédents l'adresse disque de la page concernée et réalisent l'entrée/sortie. Ils indiquent le bon fonctionnement de l'opération au programme appelant.

Le passage des paramètres se fait par registres.

Quant à la structure logique de la mémoire virtuelle d'un utilisateur, il en a été question plus haut.

Les répertoires disques :

Ils sont créés par cartes moniteur. Ils ont tous une structure en accés direct où l'enregistrement physique est le secteur. Leur contenu est conservé d'une session à l'autre et ne donne pas lieu à une phase d'initialisation.

La consultation et la mise à jour se font par deux routines utilitaires pour chaque répertoire. On distingue :

Le répertoire résumé utilisateur (RRU) : il contient le nom des utilisateurs qui ont le droit d'initialiser une session. Chaque section contient donc 32 entrées sauf le premier qui commence par une entrée indiquant le nombre total d'entrées occupées.

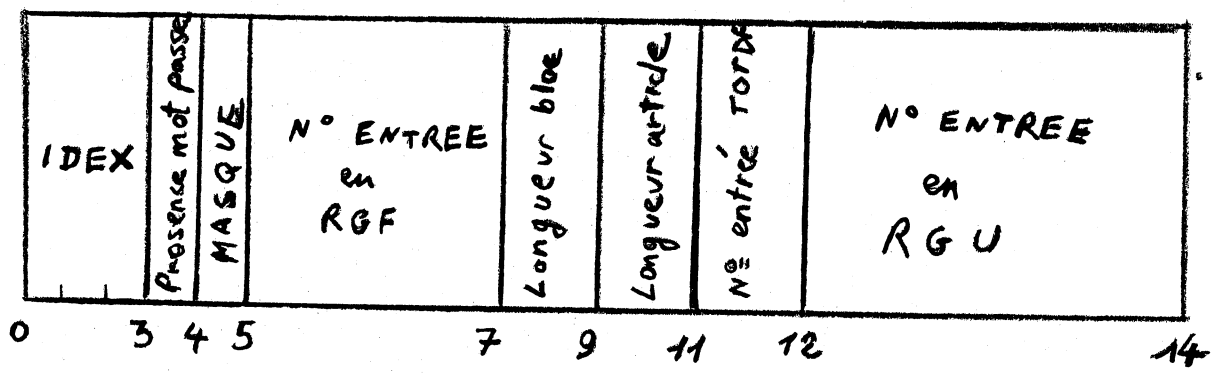
Ce répertoire est créé et initialisé en dehors d'une session. Au cours d'une session il n'est utilisé qu'en consultation par la routine utilitaire RCHRRU dont les paramètres passés par registres sont :

- en entrée le nom de l'utilisateur
- en sortie le numéro d'entrée si le nom a été trouvé et un résultat 0 si trouvé, 1 sinon.

Pour les consultations de répertoires le problème du partage d'un même fichier intervient.

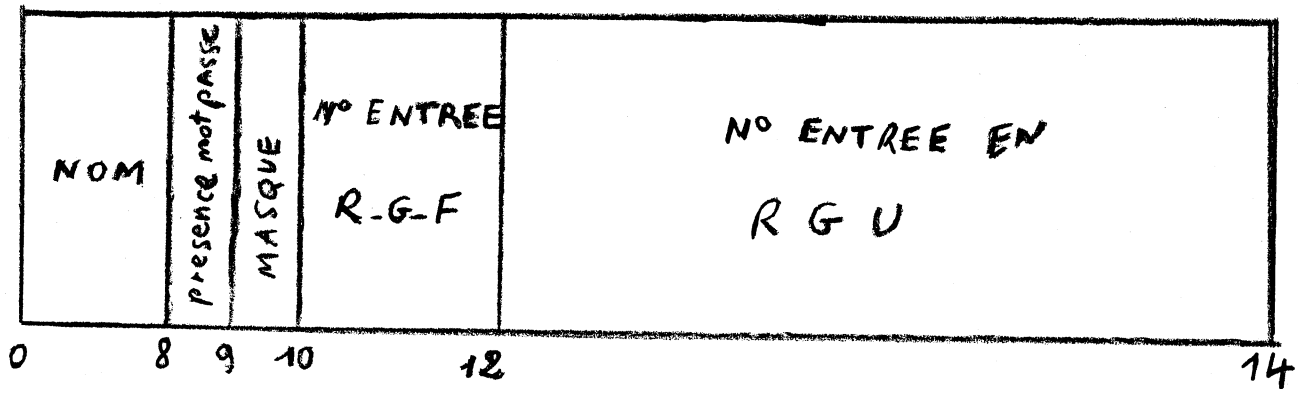
Le répertoire général utilisateur (RGU) : il contient des informations concernant l'utilisateur (1). Il est accédé en consultation au cours d'une session et l'entrée est déterminée par une consultation préalable du RRU.

Le répertoire des idex de fichiers : il contient l'idex et certaines informations concernant des fichiers de classe 3. Le format d'une entrée est le suivant :

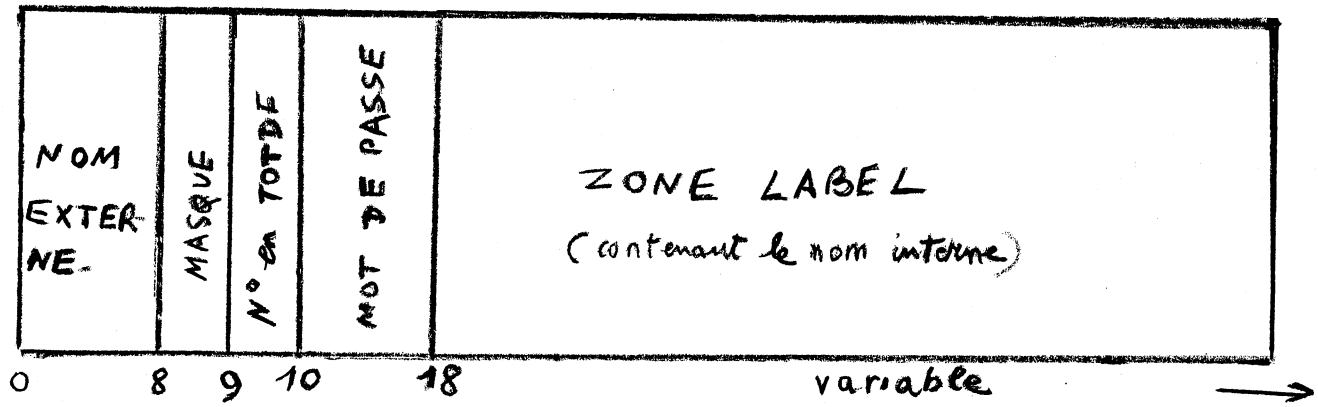


(1) cf. Travaux de Monsieur Delpuech.

Le répertoire des noms de fichiers (RNOM) : Il contient le nom et certaines informations concernant des fichiers classe 2. Son format est le suivant :



Le répertoire général fichier (RGF) : il contient des informations concernant à la fois des fichiers classes 2 et 3. Pour des raisons de simplicité chaque entrée occupe un secteur. On y trouve les informations suivantes :



Il y a deux routines utilitaires pour chacun de ces fichiers une de consultation RCHRIDX et RCHRNOM pour la consultation et RGIDX et RGRNOM pour la modification. Nous allons décrire le principe de RCHNOM et RGNOM seulement celui de RCHIDX et RGIDX étant identique. En réalité, il y a seulement deux routines RCHREP et RGREP dont les paramètres sont :

- un indicateur pour nom ou idex
- la base, le déplacement permettant de localiser l'information
- un indicateur de bon fonctionnement.

L'algorithme est le suivant : en recherche, la première entrée donne le nombre d'entrées utiles, on lit un secteur, on compare à l'intérieur entrée par entrée et on continue le mécanisme jusqu'au bout des entrées occupées. En retour, on fournit 0 ou 1 selon que l'on a trouvé ou non et on a également positionné en page 0 le déplacement dans le buffer

de lecture de la partition du début de l'entrée trouvée, ainsi que le numéro de l'entrée.

En écriture, connaissant le numéro d'entrée, on peut déterminer quel est le secteur concerné. On le lit, on remplace l'entrée et on le réécrit. Dans le cas où il s'agit d'ajouter une nouvelle entrée on commence par lire le premier enregistrement qui indique où se trouve l'entrée libre et on le réécrit une fois mis à jour à la fin.

Le fichier des messages (FMES) : ce fichier n'est pas homogène en ce qui concerne le support.

En effet, certains messages sont d'une utilisation très fréquente et sont rendus résidents pour minimiser le nombre des entrées sorties et améliorer le temps de réponse. Les messages moins fréquemment utilisés sont dans un fichier disque. Dans les deux cas, les messages sont de taille variable et précédés de leur longueur.

Le fichier disque est construit de manière statique.

Chaque message est repéré par un numéro :

- les numéros inférieurs à 1000 indiquent le rang d'un message résident et permet d'accéder au message par un index.

- pour les autres, la soustraction de 1000 au numéro donne l'adresse relative disque du message.

Une routine utilitaire GTMSG permet d'obtenir le message en tête. Pour lancer un message, il suffit ensuite de faire CALL SEND1.

Remarque :

Pour faciliter la lecture des messages disque, on a rangé un seul message par secteur.

APPENDICE 4

APPENDICE N° 4 : LES PROGRAMMES STATIQUES

On désigne sous ce vocable les programmes qui ne font pas partie de la partition conversationnelle en temps que programme.

Le dump de la mémoire virtuelle :

Il sert à connaître l'état de la mémoire virtuelle des utilisateurs sur disque. En l'associent avec le dump dynamique donnant la zone des pages, on peut savoir exactement ce qui s'est produit en cas d'erreur et le taux de remplissage des différentes tables et fichiers.

Pour des raisons d'agrément, ce programme liste les différents éléments de la mémoire d'un utilisateur en respectant le format des entrées et en faisant précéder chaque table d'un titre.

La liste se fait en format hexadécimal avec en regard l'équivalent caractères.

Un des paramètres est le numéro de l'utilisateur qui permet de sélecteur la mémoire virtuelle d'un utilisateur particulier.

Le programme de création du répertoire des messages :

Il remplit le fichier des messages résident sur disque.

Les programmes de manipulation en travail série des fichiers disque :

Ils concernent les répertoires fichier et utilisateurs. Ils permettent des mises à jour et des réorganisations en fin de session qui nettoient les répertoires en supprimant les trous. Un programme particulier permet également de supprimer physiquement un fichier qui a été marqué au cours de la session. Il libère alors l'espace disque occupé si nécessaire. Ils permettent également de créer ou de supprimer des fichiers en travail série pour les sessions suivantes.

APPENDICE 5

APPENDICE N° 5 : LISTE DES SPTC

REC2 : pour lire un enregistrement source au T.I.

SEND2 : " lister un enregistrement source au T.I.

REC2D : " lire un enregistrement donnée au T.I.

RCHUP : pour localiser en T.E. le numéro externe précédent un numéro donné de ligne source.

RCHUPD : idem pour les données.

SEND2D : pour lister un enregistrement donnée au T.I.

LISTMS : pour lister les messages dont les numéros sont dans la pile.

LISTOI : pour constituer la liste des ordres d'impression.

DF : cf. fichiers seconde partie.

DFB : " " " "

MODIF : pour modifier une ligne de numéro externe donnée.

CHGPAG : pour changer de page dans un fichier courant.

GTBUG : pour délivrer un enregistrement donnée à l'interpréteur pour un READ Fortran.

PTBUF : pour constituer un enregistrement à partir d'un WRITE Fortran.

GET2 : cf. fichier.

PUT2 : cf. fichier.

B I B L I O G R A P H I E

Auteurs

M. GRIFFITHS Analyse déterministe et compilateurs
(thèse soutenue le 25 octobre 1969 à la Faculté des Sciences de Grenoble).

E.W. DIJKSTRA "THE" Multiprogramming system.

J.A.N. LEE The anatomy of a compiler
(Reinhold computer science. Reinhold publishing corporation).

Mrs. AVON et GIRES Réalisation d'un compilateur incrémentiel et conversationnel Fortran.
(projet de fin d'étude de l'I.P.G., 1970).

S. GILL Thoughts on the sequence of writing software
(Nato Science committee Garmish 7-11 october 1968
Editors : Peter Naur and Brian Randell).

A.I. LLEWELYN and R.F. WICKENS Testing software
(idem).

E.W. DIJKSTRA Complexity controlled by hierarchical ordering of functions and variability
(idem).

Brochures

CP:CMS Manuel d'utilisation et principes de fonctionnement
M.T.S. Manuel utilisateur
Basic BULL Manuel utilisateur
BROCHURES C.I.I. : FORTRAN
ASSEMBLEUR
SGF
MONITEUR
SGT
etc ...

Brochures rédigées au cours de la réalisation du système

Spécification de définition du Fortran conversationnel
Spécification de réalisation du Fortran conversationnel
(rédigées par M. Delpuech et moi-même).